QUERY UNDERSTANDING

Applying Machine Learning Algorithms for Named Entity Recognition

IT4BI MSc Thesis

Official Supervisor:
Oscar Romero, PhD

Author:
Paul Ashaolu

Advisor:
Toni Cebrian
Softonic International S.A.

Master in Information Technologies for Business Intelligence

Universitat Politècnica de Catalunya

July 2014

UNIVERSITAT POLITÈECNICA DE CATALUNYA

Abstract

Facultat d'Informtica de Barcelona
UPC-Barcelona Tech

Master in Information Technologies for Business Intelligence

QUERY UNDERSTANDING
Applying Machine Learning Algorithms
for Named Entity Recognition

by Paul Ashaolu

The term-frequency inverse-document(tf-idf) paradigm which is often used in general search engines for ranking the relevance of documents in a corpus to a given user query, is based on the frequency of occurrence of the search key terms in the corpus. These search terms are mostly expressed in natural language thus requiring natural language processing methods. But for domain-specific search engines like a software download portal, search terms are usually expressed in forms that does not conform to grammatical rules present in natural language and as such, they cannot be tackled using natural language processing techniques. This thesis proposes named entity recognition using supervised machine learning methods as a means to understanding queries for such domain-specific search engines. Particularly, our main objective is to apply machine learning techniques to automatically learn to recognize and classify search terms according to named entity class of predefined categories they belong. By so doing, we are able to understand user intents and rank result sets according to their relevance to detected named entities present in search query.

Our approach involved three machine learning algorithms; Hidden Markov Models (HMM), Conditional Random Field(CRF) and Neural Network(NN). We followed the supervised learning approach in training these algorithms using labeled training data from sample queries, we then evaluated their performance on new unseen queries. Our empirical results showed precisions of 93% for NN which was based on distributed representations proposed by Yoshua Bengio, 85.60% for CRF and 82.84% for HMM. CRF 's precision improved to about 2% , achieving 87.40% after we generated gazetteer-based and morphological features. From our results, we were able to prove that machine learning methods for named entity recognition is useful for understanding query intents.

Acknowledgements

I would like to express my sincere appreciation to my supervisor Dr. Oscar Romero for his useful comments, remarks and guidance during the course of this thesis work. Without his encouragement and support this project would not have materialized . I would also like to thank my advisor at Softonic International S.A, Toni Cebrian. His invaluable contributions and suggestions helped me to coordinate this project.

My sincere appreciation goes to the IT4BI coordinator, Professor Esteban Zimanyi and other members of the IT4BI committee for their commitments to achieving this goal.

Special thanks goes to my friends who supported me in writing, and encouraged me to strive towards my goal. I also wish to thank the members of the Data Science team at Softonic International S.A. for there useful discussions and friendly help.

Most of all, I would like to thank my family for there constant encouragement, love and support.

Contents

List of Figures

List of Tables

This thesis is dedicated to my parents. For their endless love, support and encouragement

Chapter 1

Introduction

Information retrieval in recent times has become a global focus especially in this era of information explosion through the World Wide Web. The vast amount of data is growing daily and so also the diversified unprecedented needs of users seeking relevant information. Advances in information retrieval is helping to create new ways of retrieving relevant information from the vast amount of diverse information on the World Wide Web. This further raises a question of how to discover documents on basis of relevance to user search intent. Traditional ways in web search engines for retrieving relevant information create indexes for all documents stored in the repository. User can then search these documents through the indexes, expressing keywords representing search intent. A major challenge in this approach arises from the fact that most relevant results can only be retrieved with search keywords containing terms with which the documents have been indexed. Consequently, this requires that user search keywords must have existed in the index before the relevant results to that keyword could be fetched. Another challenge to this traditional technique of information retrieval is in the ambiguity of natural language, in which same word can capture different meanings in different context. Users communicate in natural language and as such express intentions in a similar fashion. This however has raised the awareness for a more robust technique directed towards capturing true meaning of user search intents, which is engineered towards understanding the user query.

It can be proved that query understanding on the most part can be influenced by extraction and classification of Named Entities (NE) embedded in user search words. A named entity can be defined as a word or group of words which makes reference to real world objects such as location, person, product, organization etc. The process of identifying and classifying these proper names into a set of predefined classes of interest is termed *Named Entity Recognition (NER)*. Extracting named entities from user queries

translates to extracting correct key terms in user queries which expresses query intent thus providing a thorough understanding of what the user really want.

Common practical applications of NER includes Question and Answering system, Document classification, Document searching, enrichment of information retrieval systems such as Internet search engines etc. NER techniques can also be used to recognize and classify words in a user query into predefined categories they belong in order to return the most relevant concept classes in the results, concept classes to auto-complete user query, concept classes to make correction to mis-pellings etc. Although NER has gained world wide attention of many researchers for more than a decade, the task still remains challenging, especially for domain-specific search engines which are not constrained by grammatical rules unlike the general search engines.

In general search systems, NER tasks often follow the pipeline illustrated in Figure 1.1, in which raw texts are initially pre-processed before the extraction of the named entities in them. However, for domain specific search systems, further pre-processing steps might be needed and the extraction of named entities from the pre-processed text might even require more robust technique.

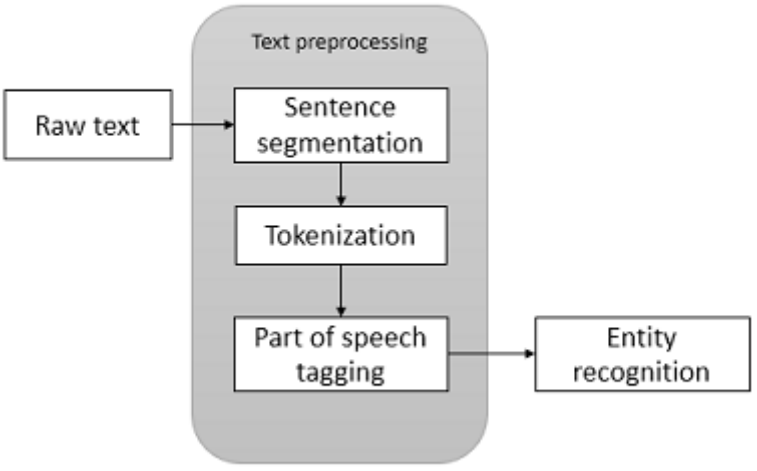

FIGURE 1.1: Named Entity Recognition pipeline

In this thesis work, we will attempt to investigate and propose a query understanding framework using machine learning techniques that are able to identify pre-defined classes of named entities mentioned in web search query texts in a domain-specific search engine.

1.1 Context of Thesis

This thesis work was carried out as a joint research project between the Information Technologies for Business Intelligence(IT4BI) consortium [1], Universitat Politècnica de Catalunya[2] and the Data Science department at Softonic International S.A.[3], both located in Barcelona. The IT4BI is an EU-funded Erasmus Mundus project between 5 leading Universities[4] in Europe. The consortium coordinates cutting edge masters and doctorate degrees in business intelligence and big data analytics. Softonic on the other hand is an internet company in Barcelona, founded in 1997. It specializes in hosting programs on dedicated servers from which users all over the world can download from. Softonic download portal is currently ranked top with an average of six million daily downloads and 120 million unique monthly visitors[5].

1.2 Motivation

With constant increase in the number of programs being uploaded on Softonic's servers and corresponding description texts about these programs, it is becoming increasingly challenging for users to search and retrieve relevant results without using specific keywords that describes desired program. The term-frequency inverse-document (Manning et al. [1]) paradigm which is often used in search engines for scoring and ranking the relevance of documents to a given user query is based on the frequency of occurrence of search keywords which are mostly expressed in natural language. But for most domain-specific search engines like Softonic download portal, search words are usually expressed in forms that does not conform to grammatical rules present in natural language and consequently cannot be tackled using techniques in natural language processing. For example, consider a user query *"need for speed android"*, which is an intention to finding an android version of an application (game) called *"Need for Speed"*. Applying natural language processing techniques such as stop-word removal for English language will eliminate "need" and "for" from the name of the program leaving only *"speed"* and *"android"* as keywords. Returned result-sets as seen in Figure 1.2 are thus composed of programs containing only these keywords in their descriptions. This is completely different from what the user intended, hence the need to explore other techniques outside the scope of natural language processing in order to calculate document's relevance to user query in this domain.

[1] http://it4bi.univ-tours.fr/
[2] http://www.fib.upc.edu/en.html
[3] http://corporate.softonic.com/
[4] Universit Libre de Bruxelles, Universit Franois Rabelais Tours, Ecole Centrale Paris, Universitat Politècnica de Catalunya, Technische Universitt Berlin
[5] As of November 2013

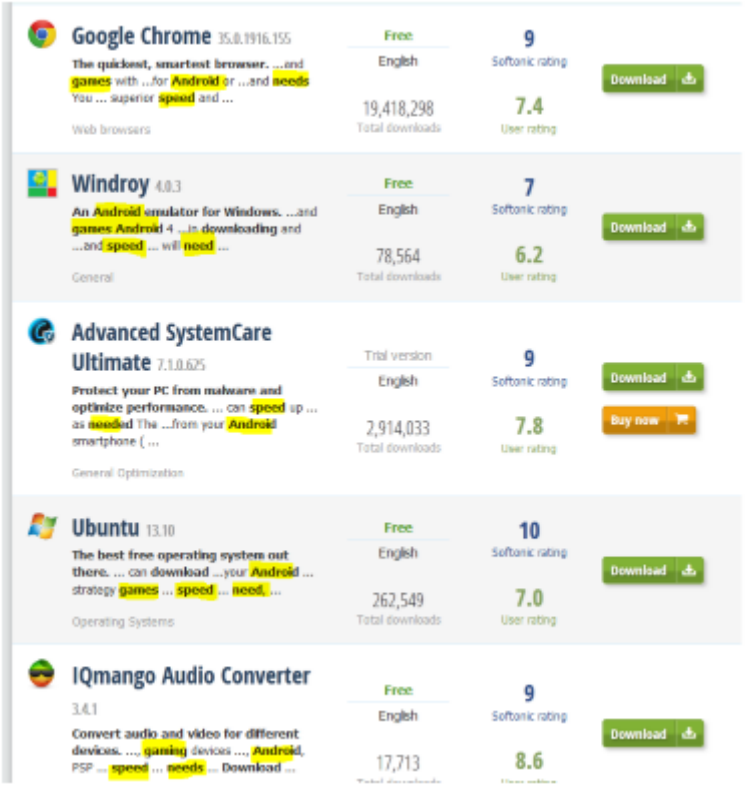

FIGURE 1.2: An illustration showing search results for the program *"need for speed android"*

Tthe Data Science team at Softonic in collaboration with the IT4BI consortium at Universitat Politècnica de Catalunya is tasked with a query understanding framework that improves the efficiency of the search engine by implementing automatic means of identifying and classifying named entities mentioned in search queries into predefined named entities such as programs, operating systems, license etc. In doing this, results matching only the identified named entities in the user search query will be returned thereby improving the quality of the search results returned to the user. This project will identify useful algorithms for this task and also experiment with real data from the historical log of search queries obtained over a particular period of time.

1.3 Problem Statement

The classification of user queries is non-trivial, the major challenge lies in the fact that queries are usually short Beitzel et al. [2] and Paşca [3] i.e. they lack context, they are ambiguous: multiple meanings depending on usage, and they lack capitalization-an important feature for identifying named entities. They reported that majority of user queries lack enough context, containing only about 2-3 terms which can also belong to multiple topical categories.

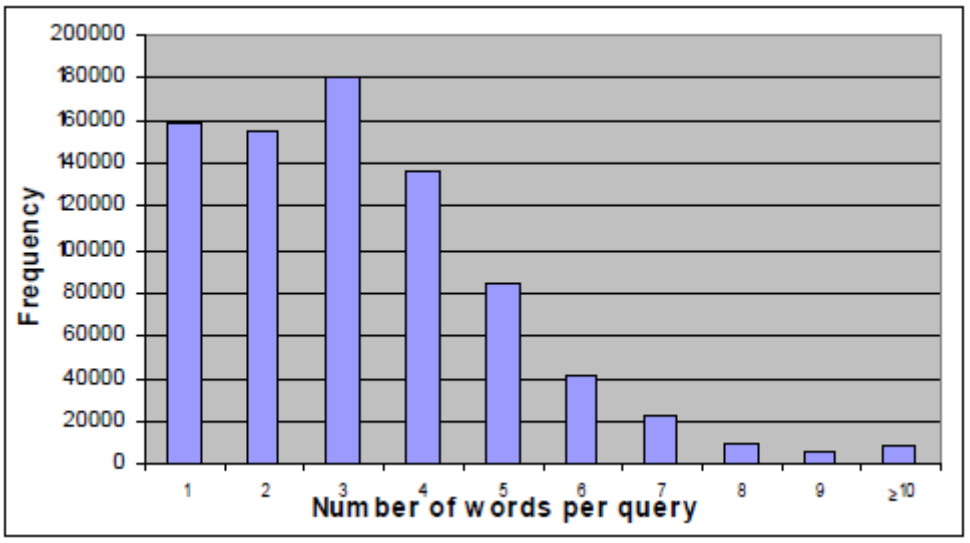

FIGURE 1.3: Frequency of word length of queries at the KDD2005

This was illustrated in the provided KDDCUP2005 data sets[6], which consists of publicly available 800,000 queries. Figure 1.3 shows the frequency distribution of query lengths and it can be seen that majority of the queries are of word length 3 (22%), which will not provide as much context as the queries with more context(word length). The queries contain variations of person's name, location, URLs and sometimes meaningless strings making it noisy. Regarding ambiguities, sometimes the word "Apple" can be expressed in a query to refer to a fruit or to the computer company.

Majority of previous studies and research in NER relied on traditional approach of information retrieval, which focuses on well-formed natural language queries. However, little progress is made regarding queries designated to retrieve specific types of information. How to understand user query in terms of a set of given target categories or taxonomies is a major research issue with considerable number of studies addressed to finding solution. Successfully understanding the information needs of users can prove beyond reasonable doubt that information providers can improve the effectiveness and efficiently of result sets of search engine systems and also help to organize and deliver personalized experience to users. While many studies and research have extensively focused on the challenges of query understanding in general search engines achieving near-human performance, as far as we know, none is focused on the domain specific to programs and software.

This thesis work presents named entity recognition techniques for understanding web search queries in the domain of computer programs and software applications. The approach is composed of machine learning models, trained with real queries from historical

[6]http://www.kdd.org/kdd-cup-2005-internet-user-search-query-categorization

datasets of past queries from users all over the world. The first stage of this work involves the collection, preparation and exploration of these datasets. After which, it will be manually annotated before being used in training machine learning algorithms for named entity recognition. Evaluation of the models will be carried out in order to access the strength and weakness of each algorithm. A dictionary-based entity recognizer was also investigated. Unlike the machine learning methods, the dictionary-based named entity recognition system does not require annotated data in order to identify named entities. However, this method requires dictionaries of named entities to be created.

1.4 Scope of Thesis

1.4.1 Objectives

Softonic International S.A. provides access to large number of programs and software applications which by searching the Softonic web portal, can be retrived and downloaded from the website. Additionally, there are other supplementary information provided to complement these programs such as articles, tutorials, advertisement of similar programs as seen on Figure1.4.

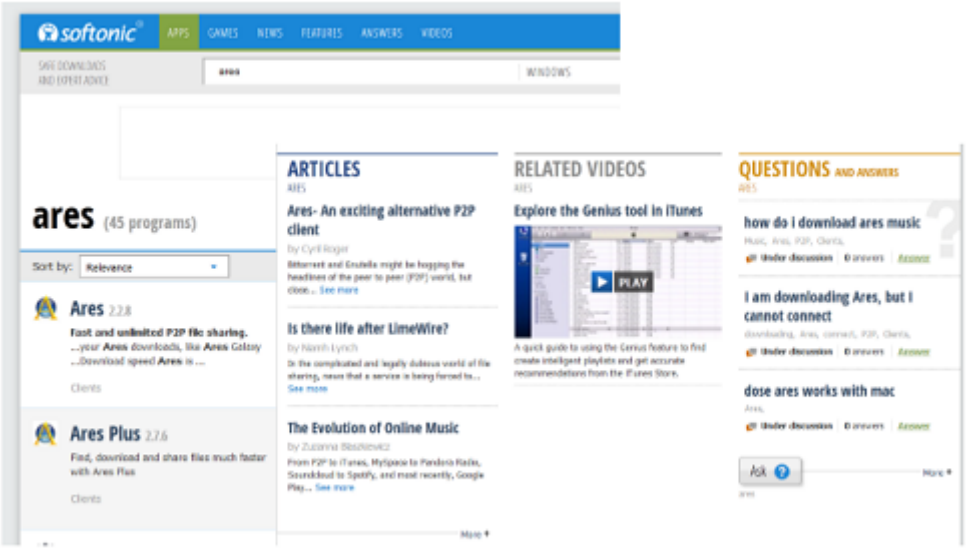

FIGURE 1.4: Search result content and articles

Large amount of information about programs increases extensively as new information about programs or related programs are added to the repository. However, it is challenging for users to identify relevant information amidst all these whenever they search

for programs. This thesis will focus on using machine learning techniques to understand users intents as expressed in the search text. Specifically in this thesis work, we will research on existing solutions for similar problems, we will develop a framework to experiment on the datasets using knowledge from our findings, and we will carry out experiment on the sample search queries using machine learning algorithms peculiar to these problems. We will document our findings and present it to Softonic and the IT4BI consortium at the end of the work.

1.4.2 Our Contribution

Our contribution to this thesis is in two folds:

1. We will create a taxonomy for named entities that captures concepts in the domain of programs and software applications.
2. We will propose a query understanding framework different from conventional natural language processing, for classifying concepts in domain-specific search systems.

1.5 Thesis Plan

Plan for this thesis work is summarized as follows:

- Study existing literature and research studies related to domain of thesis work
- Study existing algorithms and implementations related to this thesis work
- Outline algorithms to be implemented for this thesis work
- Data collection, exploration and pre-processing
- Selection of tools and platforms for experiment. Installation of tools
- Initial implementation, results, feedbacks and discussions
- Experiment setup
- Evaluation and Result discussion
- Preparation of documentation
- Presentation of thesis work

This plan is subject to change as the work progresses. Weekly meeting is scheduled with the Manager(supervisor) of this thesis in order to discuss methods and progression of this thesis work. A snapshot of the initial plan listed above can be seen in Figure 1.5:

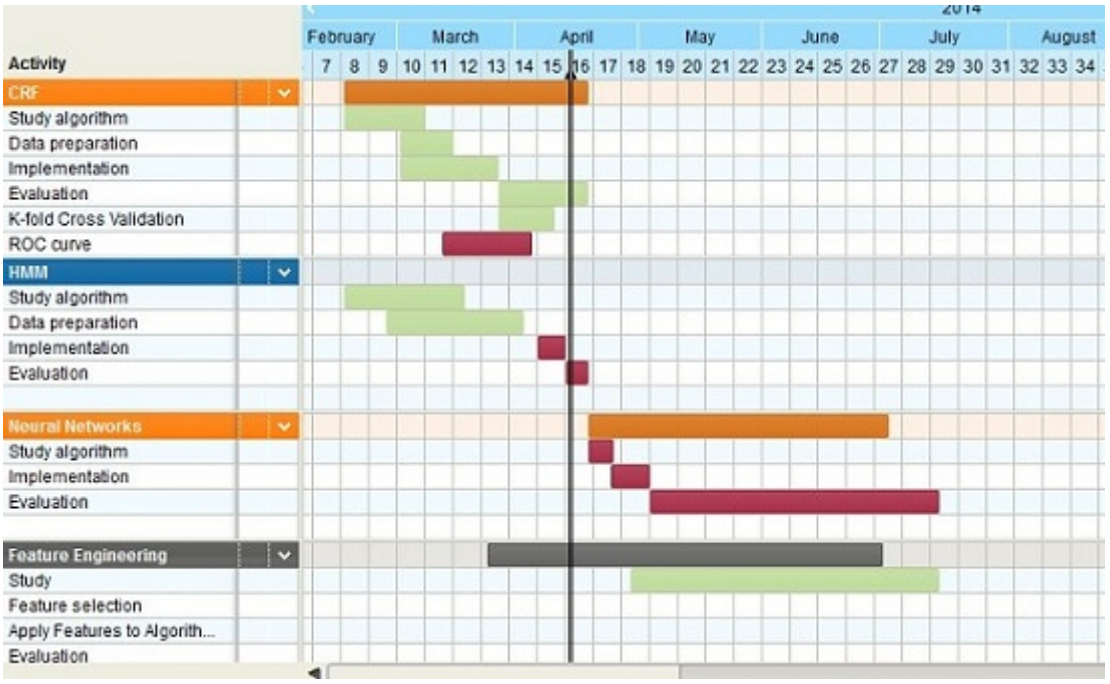

FIGURE 1.5: A snapshot of the project plan

1.6 Outline of Thesis

The structure of this thesis is organized into the following chapters as follows;

- Chapter 2: Overview of Named Entity Recognition
  This chapter introduces background information and theory of named entities and named entity recognition task.

- Chapter 3: Machine Learning methods for Named Entity Recognition
  This chapter introduces Machine learning techniques for Named Entity Recognition task, explaining different algorithms for NER, and citing some examples of usage to NER tasks.

- Chapter 4: Related Work
  This chapter presents and discusses previous research and studies conducted in understanding queries such as natural language processing methods and query classification. It also discusses the characteristics and challenges of these approach and .

- Chapter 5: Feature Engineering
  This chapter introduces feature engineering, and methods of feature selection. It also introduces distributed representation using neural networks which is a recent development in Deep Learning.

- Chapter 6: Experiment and Results
  This chapter presents the experiments; outlining the basic steps involved with training the machine learning models. Results of the evaluation of the models are also discussed in this chapter.

- Chapter 7: Conclusion and Future work
  This chapter presents the conclusion, providing an overview of the whole thesis work and the overall results. Identified opportunities for future work are also discussed.

Chapter 2

Overview of Named Entity Recognition

2.1 Named Entity

According to Grishman and Sundheim [4], the term *Named Entity* was first introduced in 1996 at the Sixth Message Understanding Conference(MUC-6) as a paradigm to identifying atomic units which belongs to a set of predefined categories in documents. Such categories include names of persons or organizations, geographical locations which at the MUC-6 competition was referred to as "enamex", and other special numeric expressions such as date, time (also known as timex), money, percentage (collectively known as numex) etc. Early studies in Named Entities mainly focused on proper names but recent research interests and enterprise needs has extended research in Named Entities into other categories across different domains e.g. Bio-sciences, where Dingare et al. [5] identified named entity *"NEWGENE"* from biomedical abstract papers , and in Geology where Sobhana et al. [6] developed a system that predicts Named Entity classes from geological text of scientific reports and articles of India. The availability of data for this type of research has further helped encourage researchers to provide solutions for named entity tasks i.e. GENIA corpus has opened up opportunities and improved interests in finding DNA, RNA and other proteins.

2.2 Named Entity Recognition

The task of Named Entity Recognition is concerned with extracting candidates named entities from textual information and classifying them into a set of predefined categories.

Named Entity Recognition task is non-trivial as there are challenges attributed to it, especially challenges common with natural language processing. Some of these challenges are rooted in ambiguity and variations of word usage in a context. For instance, *"Washington"* can be used to refer to either a location(as in Washington D.C.) or to a person's name(as in Denzel Washington) depending on the context of usage. Another common problem in NER is in word boundaries i.e. named entities usually do not occur as singular units of words rather as chunks or group of text, and as such looses semantics if not considered so. For example, *New York* must not be treated as two different words; *"New"* and *"York"* but rather as a single entity, *New-York*. Therefore there is dire need for a robust system that can correctly predict a token or group of tokens which belongs to a named entity category. Earlier studies in overcoming the problem of word boundaries(Federici et al. [7] and Sha and Pereira [8]) have suggested shallow parsing techniques to divide texts into segments which corresponds to known syntactic unit. According to Molina et al. [9], he used a specialized statistical algorithm to learn to predict word boundaries and chunk of words. The algorithm was trained using sample data labeled and encoded using the IOB format by Zhang et al. [10], which is a way of prefixing each word of an entity with a B if the word is a beginning of a named entity, or an I if it is a continuation or inside of a named entity and O if the word is outside any of the named entity.

Over the years, different approaches have been used to tackle Named Entity Recognition. Broadly speaking, the most common approaches can be categorized as hand-crafted rule-based, Dictionary-based techniques and machine learning approach. Among these categories there are sub-approaches that collaborate to overlap with some categories at the top-level. For example, some researchers have automatically generated candidates for Dictionary using machine learning (Banko et al. [11]), while some used Dictionary information as a feature in Machine learning based Named Entity Recognition (Kou et al. [12])

2.2.1 Rule-based approach

Rule-based NER systems utilize hand-crafted pattern-matching rules to derive heuristics about the morphology and the semantics of input texts in order to identify and classify according to the named entity class membership. This method basically relies on the intuition of the designer who constructs a set of rules perceived to capture the notion of named entities in a given text. The rule-based basically looks at strings or the surrounding strings of named entities to be classified. This approach appears to be complex since it requires many rules in order to cover the particular domain therefore lacking robustness and making cross-domain portability impossible. New rules had to

be made for every new source. For example a rule for identifying street addresses in the UK can be crafted to follow a set pattern, and this will be different from the pattern of street addresses in Spain. Other variations of street address in another country has to be catered for using new set of rules.

The rule-based technique is suitable for domains with constraints but suffers greatly from its inability to adapt to new domains. Common usage has been as a standalone rule based Named Entity Recognition system (Farmakiotou et al. [13]) or in combination with other Named Entity Recognition approache as classifiers in machine-learning approaches (e.g. Abdallah et al. [14] and Gali et al. [15]), or as candidate taggers in Dictionary match techniques(Kou et al. [12]).

2.2.2 Dictionary-based

Dictionary approach is based on using knowledge-base sources. Text are matched against a constructed gazetteer or dictionary in order to classify the text according to the named entity class of the gazetteer. Using this approach, one has to either manually or dynamically create a dictionary from a corpus. This approach to NER can be daunting especially when the dictionary becomes outdated, a new dictionary have to be manually created. However, dictionary-based approach typically have a low recall but the quality can be improved by combining it with soft-matching rules which can resolve multiple names to the same entity. Kou et al. [12] followed this approach by coupling a protein dictionary with a hidden markov model (HMM) to implement soft matching of phrases to entries in the dictionary. Other examples of dictionary-based approach for named entity recognition can be seen in the works of Kazama and Torisawa [16], who used Wikipedia as external knowledge in recognizing named entities at the CoNLL 2003 shared task[1], and Bravo et al. [17] who explored fuzzy matching techniques for identifying Biomedical named entities from a curated gene and disease dictionary.

2.2.3 Machine learning approach

Machine learning Named Entity Recognition is based on converting Named Entity Recognition problems to classification problems and then using statistical learning algorithms and some feature representation to make predictions about named entities in texts. Basically, they search for patterns or relationships in texts and then automatically construct rules for statistical and machine learning algorithms to learn from, in order to

[1]Conference on Computational Natural Language Learning held in Edmonton, Canada at May 31 and June 1, 2003

generalize over new unseen texts. Machine learning techniques in Named Entity Recognition can be broadly classified into two categories: Supervised and Unsupervised. An hybrid of these two categories is the Semi-supervised machine learning, which combines labeled and unlabeled data for inductive learning.

Machine learning methods are robust and trainable which makes them adapt to new domains unlike the rule- and dictionary-based approach. But they might require large amounts of annotated or labeled data (in case of supervised learning) in order to achieve good performance. In recent years, different machine learning algorithms have been proposed for NER tasks, Bikel et al. [18], Borthwick et al. [19] amongst others proposed Hidden Markov Models for finding proper names from text corpora, Bender et al. [20] and Chieu and Ng [21] used Maximum Entropy Models (MEM), McCallum and Li [22] used Conditional Random Fields, for the CoNLL-2003 shared task. Recent achievements in computational capacity of computer systems has also benefitted machine learning systems in solving large-scale NER experiments and complex NER problems (Cucerzan [23]).

Chapter 3

Machine Learning methods for Named Entity Recognition

3.1 Machine Learning and Language Models

A language model is a function which learns characteristics of word distribution over a sequence of words i.e. given a sequence of words such as a sentence, a language model is able to predict next word given the previous words in the sentence. The NER task can be modeled as a sequence labeling problem in which the task is to assign each word token in a given sequence to a label drawn from a finite set, where there exist inter-dependent relationships between the labels of each word token in the sequence. Machine learning which evolved from artificial intelligence, has been widely used in sequence labeling tasks in different domains to learn the precise assignment of labels to sequence of words. Mitchell and Michell [24] described machine learning as the development of algorithms that automatically optimizes performance measures using information gathered from past experience. This is possible because machine learning applies theories in statistics to develop mathematical models that are capable of making inductive inference based on the given sample information.

The dynamics of machine learning approach to language modeling tasks uses algorithms to learn patterns and relationships in texts or textual documents for improving prediction or inferential performance. Machine learning language modeling can be broadly categorized into two; *supervised learning* and *unsupervised learning*. Supervised learning involves using labeled examples to teach a model to infer mappings from input x to output y, guided by $x - y$ example pairs in the training set. These labeled examples are expected to be labeled by domain experts in order to ensure the mapping function learns the correct distinction between the target classes. Unsupervised learning on the other

hand relies on the system to learn similarities in the context using statistics gathered from the unlabeled data in order to identify groups of instances i.e. by clustering similar entities and words together. Semi-supervised learning is in between supervised and unsupervised learning, in which a machine learning algorithm collaborates knowledge and learns from both labeled and unlabeled textual examples.

3.1.1 Supervised Named Entity Recognition

Over the years, supervised machine learning has proved to be successful in identifying named entities as reported in the Message Understanding Conference (MUC) NER task (Bikel et al. [25]). However, the challenges lie in the difficulty of acquiring the required large number of manually labeled examples from which the model derives the mappings from, and also the need to manually prepare a new set of training examples for a new problem even if the algorithm remains the same. In standard supervised NER, the labeled examples is required to consist of word-label pairs drawn independently from a distribution of word-labels. A test set is also drawn to evaluate the performance of the learning algorithm.

In supervised NER, the objective of the learning algorithm is to find a function that can model the mappings from word to label, where a loss function evaluates the loss or error measure of the concensus between the predicted and expected label. During the learning process, the training examples help to minimize this error measure evaluated on the test set (error minimization) by incrementally tuning the parameter of the model in order to optimize the objective function on the training set. In general terms, the most common form of evaluating the quality of a learning process is in its ability to generalize over new unseen instances drawn from the same distribution as the training examples. Standard metrics of evaluation includes Precision, Recall and F-score. These will be further discussed in Chapter 6.

Some examples of using supervised learning algorithms used in NER tasks includes Settles [26] who used Conditional Random Field for NER in the domain of Biosciences, Su et al. [27] and Mayfield et al. [28] who used Hidden Markov Model and Support Vector Machines respectively for the English MUC-6 and MUC-7 NER Shared tasks.

3.1.2 Unsupervised Named Entity Recognition

In unsupervised NER, representations are built for unlabeled data usually by clustering similar documents or entities together. These representations can then be used to classify words into target labels. Other unsupervised techniques rely on seed rules or

on lexical resources such as WordNet for input texts to acquire cues about the context which can then be extended to classify words into target labels. Collins and Singer [29] introduced an algorithm for learning from unlabeled examples using simple seed rules based on spellings e.g. *(Mr.) indicates next word is PERSON*, and contexts e.g. *David, the king of . . . , "king of" indicates proper name "David" is PERSON*

Unsupervised learning have also been combined with supervised learning approach in NER tasks, as reported by Buchholz and van den Bosch [30].

3.2 Statistical Language Models

Attempts at using statistical language models for modelling inter-dependencies between probability distribution in a word sequence is often difficult and challenging. Typically, languages are infinite making it impossible to assign a probability distribution over all possible sentences. On the other hand, treating sentences as a sequence of words gives the possibility to decompose a sentence probability into a product of the probabilities of the words in the sequence. According to Chain rule, the joint probability of a sequence of events $x_1, x_2, x_3 \ldots, x_n$, can be expressed as a chain of conditional probabilities;

$$p(x_1, x_2, x_3, \ldots, x_n) = p(x_1)p(x_n|x_1, x_2, x_3, \ldots, x_{n-1}) \tag{3.1}$$

Earlier methodologies in language models revolved around N-gram language models, in which a learning algorithm simply counts the co-occurence of words in different lengths. The intuition is that the probability of a word occurrence w can be estimated as a function of the frequency of co-occurrence of immediate preceding word H seen in the training set i.e.

$$p(w|H) = \frac{C(Hw)}{C(H)} \tag{3.2}$$

Where $C(Hw)$ is the frequency count of Hw sequence occurrence in the training data and H is the context which consists of words depending on the n-gram. For a unigram $|H| = 0$, bigram $|H| = 1$, trigram $|H| = 2$ etc. For an unseen sequence, a smoothing technique such as "add-one smoothing" is required. N-gram model is noted for its speed in terms of performance, but suffers a bottleneck in learning from longer context, the n-gram model grows exponentially as the length of the context increases. In machine learning, the task of NER in sentences is formulated as a sequential labeling task in which the objective is to assign a sequence of labels drawn from a finite alphabet of named entities, to a sequence of input data guided by a learning algorithm. The algorithm learns the conditional distribution of next words given the preceding words.

Generally, standard classification problem is assumed to be independently and identically distributed (i.i.d.), but this is not the case in NER because words in sentences or letters in words are significantly correlated. The order of words in a sequence are constrained structurally by grammatical rules in order to convey meaning, instead of a random combination. Therefore, it is reasonable to assume the presence of sequence boundaries. Considering this, a machine learning approach for sequence labelling task is formalized as follows:

Let (x_i, y_i) for $i = 1$ to N be a set of N training examples, where each example is a pair of sequences (x_i, y_i), $x_i = (x_{i,1}, x_{i,2} \dots x_{i,T_i})$ AND $y_i = (y_{i,1}, y_{i,2} \dots y_{i,T_i})$. The goal is to construct a classifier h which can correctly predict a new label sequence $y = h(x)$ given an input sequence x

Classical probabilistic graphical models represents sequential variables using nodes and probabilistic relationship between nodes as edges. This makes it possible to express the joint probabilities over all of the sequence of variables as a factor which depends only on a subset of the variables. The Naive Bayes Model approaches this by assuming that input variables are conditionally independent to each other. This assumption is known as the Naive Bayes assumption by Hand and Yu [31] i.e. Given an input vector $\vec{x}$ and class variable y, The conditional probability distribution according to Bayes law is given as:

$$p(y|\vec{x}) = \frac{p(y)p(\vec{x}, y)}{p(\vec{x})} \tag{3.3}$$

where $\vec{x}$ in the denominator acts as a normalization constant which can be calculated by considering all possible values of y. The numerator can thus be expressed as a joint probability:

$$p(y)p(\vec{x}, y) = p(y, \vec{x}) \tag{3.4}$$

In practice, this can be too complex to compute. Using the chain rule, the joint probability can be decomposed into:

$$p(y, \vec{x}) = p(y) \prod_{i=2}^{m} p(x_i|x_{i-1}, \dots, x_1, y) \tag{3.5}$$

Recall the Naive Bayes assumption, $p(x_i|y, x_j) = p(x_i|y)$ for all $i \neq j$. The Naive Bayes classifier can thus be expressed as:

$$p(y|\vec{x}) \propto p(y, \vec{x}) = p(y) \prod_{i=1}^{m} p(x_i|y) \tag{3.6}$$

This leaves out the dependencies between the variables of $\vec{x}$ and thus is non-applicable to prediction of sequence labels in real-life situations because in real word scenarios,

words in a sequence depend on each other. However, Naive Bayes has been used for other real-life tasks such as email classification (Kiritchenko and Matwin [32])

3.2.1 Hidden Markov Model

The Hidden Markov Model(HMM) by Rabiner [33] is an extension of Naive Bayes Model, it models dependencies between variables of $\vec{x}$ making it suitable for predicting sequence of labels. HMM predicts by maximizing joint probability distributions over sequence of class labels $\vec{y}$ for a sequence of observations $\vec{x}$. That is, given a sequence observation $\vec{x} =$

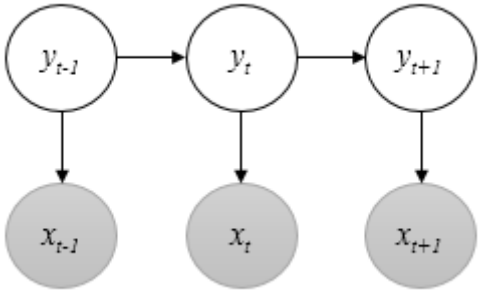

FIGURE 3.1: Hidden Markov Model

$(x_1, x_2 \ldots, x_n)$, an HMM model learns a generative model over the given observations by computing the maximum likelihood of label sequence $\vec{y} = (y_1, y_2 \ldots, y_n)$ that maximizes $p(\vec{y}|\vec{x})$ i.e.

$$\vec{y} = \arg\max_{y} p(\vec{y}|\vec{x}) \tag{3.7}$$

which can be fully expressed as:

$$\vec{y} = \arg\max_{y} \left(\frac{p(\vec{y})p(\vec{x}|\vec{y})}{p(\vec{x})} \right) \tag{3.8}$$

This joint probabilities between $\vec{x}$ and $\vec{y}$ is not tractable, as the size of $\vec{y}$ can grow exponentially. So, in order to be able to compute the probability distributions, HMM makes two independence assumptions as illustrated in Figure 3.1:

1. Each label y_i depends on a previous label y_{i-1}, this is known as Markov property
2. Each observation x_i depends on the current label y_i

These assumptions help to reformulate the joint probability of $\vec{y}$ with $\vec{x}$ as :

$$\prod_{i=1}^{n} p(x_i|y_i)p(y_i|y_{i-1}) \tag{3.9}$$

where the distribution $p(y_i|y_{i-1})$ also known as state transition probabilities, determines how previous labels are related and $p(x_i|y_i)$, known as Emission probabilities determines how the observed values of x are related to the hidden values of y. HMM is represented by three parameters: π, A and B which is given as:

1. π: Initial distribution $p_0(y_i) = p(y_i|y_0)$ i.e. probability of label y to begin a sentence i.
2. A: State transition probabilities $p(y_i|y_{i-1})$ i.e. probability to go from one state $i-1$ to the next state i.
3. B: Emission probabilities $p(x_i|y_i)$ i.e. occurrence of word x_i in position y_i

and

$$\theta = (\pi, A, B) \tag{3.10}$$

Training an HMM model means estimating parameters θ using maximum likelihood on labeled training examples in a forward-backward algorithm. The Forward-backward algorithm is an algorithm that can estimate marginal probability for each distinct state. Predicting state sequence y given observation sequence x can thus be expressed as inferring the most likely state sequence for the given observation sequence. This inference is based on adjusting the HMM parameters θ to a minimal loss function L. The loss function measures the deviation between the prediction and the desired output y. The prediction task can then be expressed using an arbitrary value of L:

$$\bar{y} = \arg\min_z \sum_y p(y|x)L(z, y) \tag{3.11}$$

Computing K number of labels for sequence size T requires $O(K^T)$ time complexity. With the loss function, the complexity can be reduced to $O(K^2T)$ under two conditions:

1. The condition where the loss function is taken over all the sequence. By using dynamic programming such as the Viterbi algorithm(Forney Jr [34]), the label y with maximum $p(y|x)$ is computed. Viterbi computes for each label at a time step, the probability of the most likely path starting at time 0 ending at time t with a label. At the end of the sequence, the algorithm would have computed the most likely path and its probability.
2. The condition where the loss function is fragmented into different resolutions for each value of y. The Forward-Backward algorithm(Fan [35]) can be used to compute the marginal probability for each y. This is done in two stages: a left to right

pass which fills a table of $\alpha_t(y_t)$ denoting $p(y_1, \ldots, y_t|x_1, \ldots, x_t)$ and a right to left pass, which fills a table $\beta_t(y_t)$ denoting $p(y_1, \ldots, y_{T_i}|x_{t+1}, \ldots, x_{T_i})$. The desired probability which is the predicted y that minimizes the loss function, denoted as $p(y_t = u|x)$ can then be derived from:

$$\gamma_t(u) = \frac{\alpha_t(u) \cdot \beta_t(u)}{\sum_v \alpha_t(v) \cdot \beta_t(v)} \tag{3.12}$$

HMM is simple and easy to train in its approach to sequential labeling, but it suffers a limitation in its representation of the true nature of sequential data due to its independence assumptions. In reality, characteristics of surrounding objects in a sequence has an influence on each objects in the sequence. For example, Dietterich [36] established that the relationship between two labels in a sequence must be expressed through the intervening labels, and the first-order Markov model where $p(y_t)$ only depends on y_{t-1} cannot capture this kind of relationship. Studies have explored possible solutions to this in HMM by proposing conditional algorithms such as Input-Output HMM (IOHMM), Maximum Entropy Markov Model (MEMM), Conditional Random Field (CRF) etc. However, HMM reportedly has recorded good performance in various NER tasks especially in the CoNLL-2003 Shared Task for Named Entity Recognition: Tjong Kim Sang and De Meulder [37], Florian et al. [38] and Mayfield et al. [28]

3.2.2 Conditional Random Field

Conditional Random Field (CRFs) was introduced by Lafferty et al. [39] to overcome the challenges posed by the independence assumption attributed to HMM for sequential labelling. Relationships in adjacent labels y are modeled as Markov Random Fields such that their relationship is conditioned on the value of inputs, x i.e. the input features of x determines the relationship between adjacent labels y. In general, CRF is an undirected graphical model that factorizes in such a way that conditionally independent nodes are not in the same scope. As illustrated in Figure 3.2, each node in CRF corresponds to variables in which their distribution is to be inferred. The edges on the other hand denotes the relationships between the random variables. CRF models are trained to maximize the conditional probability of state sequences for observation sequences i.e. $\vec{y} = \arg\max_y p(\vec{y}|\vec{x})$ unlike in HMM which maximizes the joint probabilities between sequences of labels and observations.

The conditional probability of a state sequence $\vec{y} = (y_1, y_2, \ldots, y_i)$ given the observation sequence $\vec{x} = (x_1, x_2, \ldots, x_i)$ can be defined as:

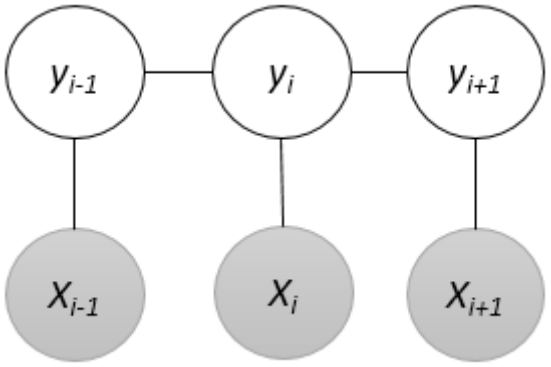

FIGURE 3.2: Conditional Random Field

$$p(\vec{y} \mid \vec{x}) = \frac{1}{Z(\vec{x})} \exp\left(\sum_{t=1}^{i} F(\vec{y}, \vec{x}, t)\right) \tag{3.13}$$

where $Z(\vec{x}) = \sum_y \exp\left(\sum_{t=1}^{i} F(\vec{y}, \vec{x}, t)\right)$ is a normalization constant also known as partition function, summing over all the labels, and $F(\vec{y}, \vec{x}, t)$ is the sum of the features at position time t, given as:

$$F(\vec{y}, \vec{x}, t) = \sum_i \lambda_i f_i(y_{i+1}, y_i) + \sum_j \lambda_j g_j(\vec{x}, y_t) \tag{3.14}$$

Where f_i and g_j are the edge and node features respectively, while λ_i and λ_j are their corresponding feature weights. The edge features expresses the sequential relationship between adjacent labels y_{i+1} and y_i, while the node features defines how properties of the observation data affects the label prediction at time t. The value of the feature functions are usually in binary, for example,

$$F(\vec{y}, \vec{x}, t) = \begin{cases} 1, & \text{if } y_{i-1} = \text{JJ}, y_i = \text{NN} \wedge x_t = \text{ending "-tion"} \\ 0, & \text{otherwise} \end{cases}$$

The following feature function captures the sequential dependency between continuous labels JJ (adjective) and NN (noun), and the current observation word ending with "*ing*". In practice, rare features tend to degenerate the performance of the CRF model, therefore there is a need to select only features with occurrence above a certain threshold. This is known as *Feature Selection*, various methods of doing this is discussed in Chapter 5. Predicting the most likely label sequence $y\prime$ given the observations can then be expressed as:

$$y\prime = \arg\max_{y} p(\vec{y} \mid \vec{x}) = \arg\max_{y} \left(\exp\left(\sum_{t=1}^{i} F(\vec{y}, \vec{x}, t) \right) \right) \tag{3.15}$$

This can be computed using dynamic programming techniques such as Viterbi algorithm. Training a CRF model using supervised learning on the other hand means finding the parameters $\theta = (\lambda_1, \lambda_2, \ldots)$ that maximizes the log-likelihood, L, given training examples $\{x^i, y^i\}_{i=1}^{N}$ i.e. finding the optimal parameters which the training data support best:

$$L = \sum_{i=1}^{N} \log\left(p_\theta(y^i \mid x^i)\right) \tag{3.16}$$

which can also be expressed as:

$$L = \sum_{i=1}^{N} \log\left(p_\theta(y^i \mid x^i)\right) = \sum_{i=1}^{N} \sum_{t=1}^{T} F(y^i, x^i, t) \tag{3.17}$$

by regularizing the training i.e. adding Gaussian prior, the equation becomes:

$$L = \sum_{i=1}^{N} \sum_{t=1}^{T} F(y^i, x^i, t) - \sum_{i=1}^{N} \log Z(x^i) \tag{3.18}$$

Numerical methods of estimating θ involves the evaluation of the log-likelihood and the gradient, that is using iterative approach to improve log-likelihood in a stepwise manner. Typically, this is done using quasi-Newton methods like Limited-memory BroydenFletcherGoldfarbShanno (L-BFGS) algorithm (Nocedal [40]), which is an iterative procedure for solving non-linear optimizations. Other similar methods for doing this include Stochastic gradient (Vishwanathan et al. [41]), which updates the parameters upon seeing one instance of data, Voted Perceptron by Rosenblatt [42], which first tries to predict label sequence y for an instance of observation, and then checks the error before updating the parameters.

Although CRF provides better prediction than HMM, memory management and training time is a challenging issue. For example, L-BFGS during training can keep up to 10-20 gradients and also takes around 100-1000 iterations in order to converge. Inspite of these challenges, CRF has been reported to perform better than HMM on sequential labelling tasks such as hand-writing recognition, named entity recognition etc. He and Kayaalp [43], McDonald and Pereira [44] used CRF techniques to identify mentions of biological named entities such as proteins and genes, Zhao and Liu [45] used CRF to

identify product named entities from Chinese text, Feng et al. [46] used CRF to identify hand writings from historical documents, etc.

3.2.3 Neural Networks

Artificial Neural Networks was developed as a computational model of the human brain. It consists of interconnected artificial neurons called nodes, organized in layers(input, hidden and output layers) and processing information via the many interconnections. The basis of the artificial neural network is the artificial neurons proposed by McCulloch and Pitts [47] (see Figure 3.3), which computes the weighted sum of given inputs and then produces a binary output. The binary value of the output depends on a certain threshold such that it is 1 if the weighted sum is above the threshold, and 0 if otherwise:

$$f(x) = \begin{cases} 1, & \text{if } x > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3.19}$$

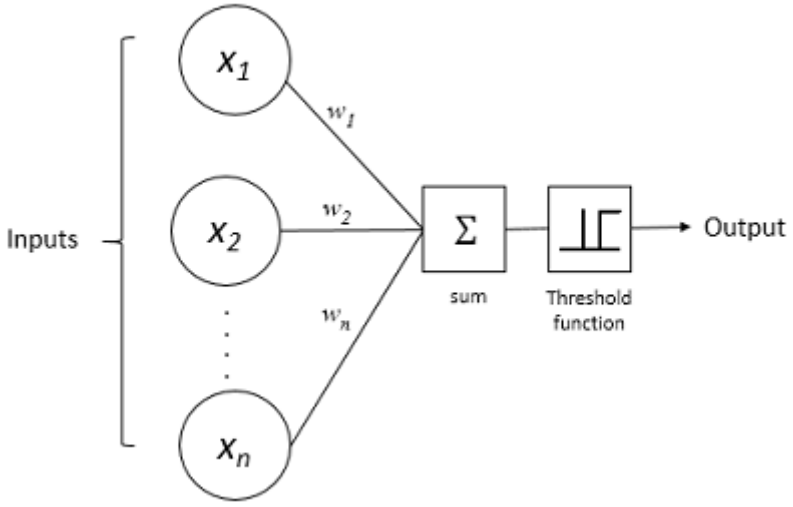

FIGURE 3.3: McCulloch and Pitts [47] Model of an artificial a neuron

However, this model was inadequate to model the true behaviour of biological neurons which are able to do non-linear summation of inputs, therefore the threshold function was replaced by an activation function which can convert the activation level of a given neuron into an output signal. Figure 3.4 shows shows some activation functions in Neural Network.

Artificial Neural Network according to the connection pattern can be categorized into two:

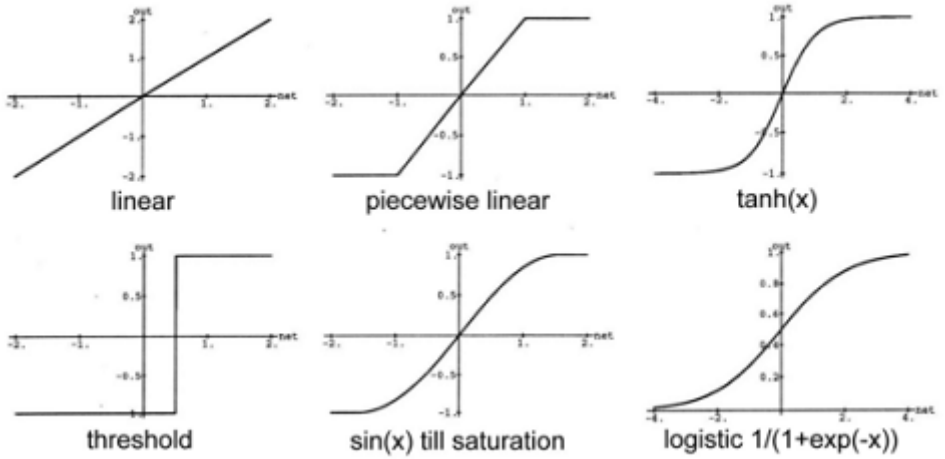

FIGURE 3.4: Neural Network Activation functions, Graves [48]

1. Feed-forward network, which is characterized by its layered architecture where each layer consist of one or more artificial neurons or nodes connected to nodes of other layers via real-valued weights as shown in Figure 3.5. Inputs are independent of previous state i.e. output of any layer has no effect on that same layer.

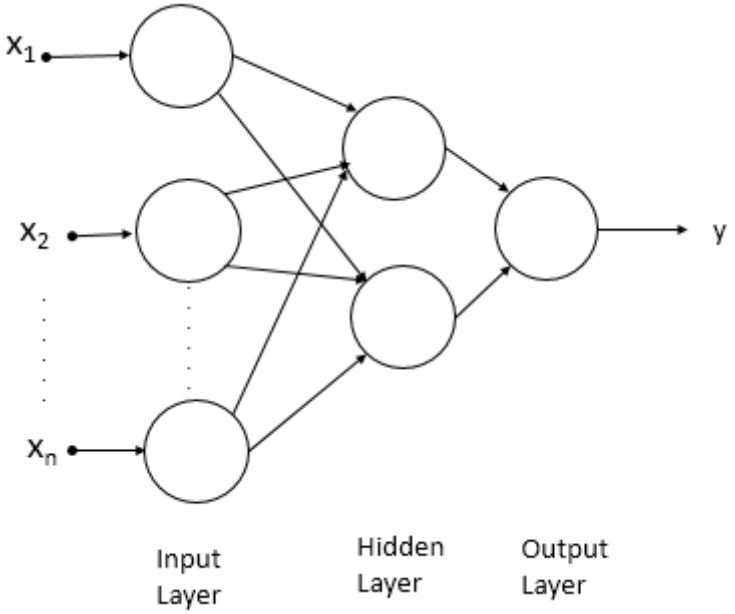

FIGURE 3.5: A Feedforward Neural Network architecture

2. Recurrent network, a network of neurons composed of feedback connections or loops as seen in Figure 3.6. Recurrent networks are dynamic, the state changes continuously until an equilibrium point is reached.

The supervised learning paradigm in Multilayer Neural Networks rely on using input

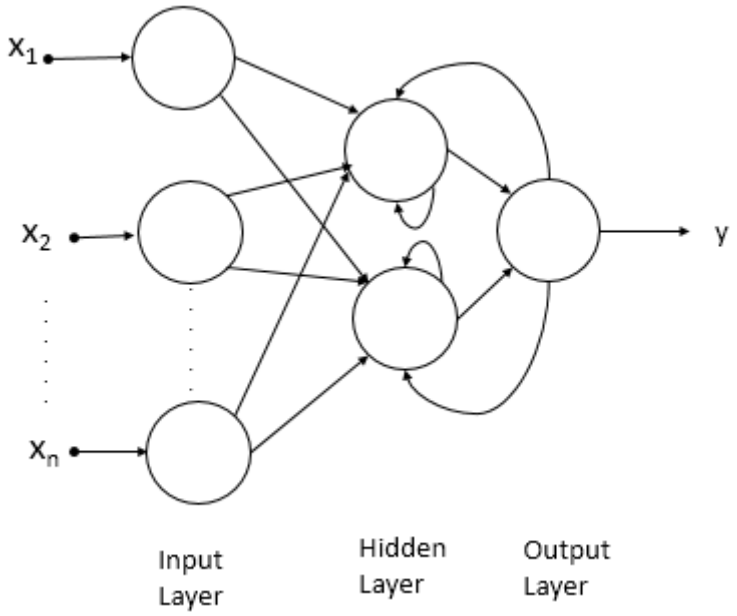

FIGURE 3.6: Recurrent Neural Network architecture

data of known output to train the network such that error signal between actual output and the desired output gradually adjust the connection weights until the error is minimized. This method is referred to as *Back-propagation*. The Back-propagation algorithm computes the minimum value of the error function via a technique known as gradient descent. For example, let

$$z_j = \sum w_{ij} x_j + \theta_j \tag{3.20}$$

be the weighted sum of input to node j^{th} in the hidden layer, where θ_j is the bias (value 1) and O_j be the activation function (e.g. sigmoid given as $\frac{1}{1+e^{-z}}$). The difference between the actual output and the expected output is given as:

$$\Delta = (T_k - O_k) \tag{3.21}$$

where T_k is the expected output and O_k is the actual value at the output node k. The Error signal at output k is then calculated as:

$$\delta_k = \Delta_k O_k(1 - O_k) = (T_k - O_k)O_k(1 - O_k) \tag{3.22}$$

where $O_k(1 - O_k)$ denotes the derivative of the activation function. The weight $w_{j,k}$ between the input node k and output node j is adjusted using δ_k:

$$\Delta w_{j,k} = L_{rate}\delta_k x_k \tag{3.23}$$

where L_{rate} is the learning rate which shows the relative change in the weight. Usually, L_{rate} is kept small in order to prevent the network from oscillating around the minimum and consequently miss the minimum point. The weight is thus updated:

$$w_{j,k} = \Delta w_{j,k} + w_{j,k} \tag{3.24}$$

3.2.4 Neural Network Language Models

In neural networks, input and output variables can only be in numeric format. In order to be able to use Neural Network for texts, input and output word tokens needs to be represented by their feature vector. The feature vector is expected to contain the semantic and grammatical interpretation of the word it represents, therefore they can also be called word feature or word representation. According to Turian et al. [49], word feature is defined as a mathematical object associated to words in which each dimension in a representation that corresponds to a feature.

In natural language processing, input representation is done by mapping words to a vector representation. The vector representation is stored in a word embedding. Traditional ways of inducing word representations often involve using supervised NLP lexicons to convert words into symbols which can then be changed into feature vector using one-hot representation. However, one-hot representations are sparse vectors generated from training examples and thus, they are difficult to generalize over on-seen data. This challenge has attracted the interest of many researchers over the years and majority have proposed unsupervised learning as a means of inducing word representations. Among these methods are word representation induced from hierarchical clustering by Miller et al. [50], distributional word representation induced from Latent Semantic Analysis by Dumais et al. [51], Latent Semantic Indexing and Latent Dirichlet Allocation by Blei et al. [52] and Neural Network Language model-induced distributed representations by Bengio [53]. Earlier work done by Hinton [54] on Neural Network to learn distributed representation of words was based on two criteria: (1) The definition of concepts is derived from its relationship with other concepts i.e. input to a Neural Network can be a single unit of concept, the connections between units can be used to encode the relationship between these concepts. (2) Concept represents set of features in which

each feature can serve as a unit, and the connection between the units represents the relationship.

The Neural Network Language Model (NNLM) which is an extension of the work of Hinton [54], was introduced by Bengio [53] such that Neural Network techniques are used to teach a language model the distribution of words and the probability function of word sequences which are semantically close to each other according to the representations. The strength of the NNLM in prediction relies on its ability to model continuous variables or distributed representations needed to improve generalizations over unseen word sequences. The N-gram models, being a local representation incorporate the specific word forms of a sequence as features and therefore the number of features needed to capture the possible sequences grows exponentially with the sequence length as mentioned in the previous section. However, as humans, we would probably tend to select semantical features to characterize words (gender, number, animation, humanness, etc.). These can be dependent on a longer range context of a word and are not mutually exclusive. They form a so-called distributed representation that can be learned by an NN. In distributed representation, N-gram model is projected onto a continuous space and a NNLM is trained to learn this continuous projection and estimate the probabilities within this projection, thus it is able to naturally handle unseen contexts.

Although, the n-gram language model probability is achieved by table lookup operations, the computation of word probabilities in NNLM on the other hand requires a complete forward pass in a NN which is expensive for large networks.

3.2.5 The mathematics of the Neural Network Language Model

Recall equation 3.1, common methods of approximating $p(x_n|x_1, x_2, \ldots, x_{n-1})$ are based on using fixed size context e.g. $n-1$ as seen in n-gram models. The NNLM model on the other hand learns by mapping each word in the context x_{n-1} in vocabulary V, to an associated d-dimensional feature vector $C_{x_{n-1}} \in \mathbb{R}^m$ of a parameter matrix C equivalent to $|V| \times m$ matrix of free parameters. where $\vec{C_k}$ contains learned features of word k and $\vec{x}$ is the concatenations of all the $n-1$ feature vectors:

$$\vec{x} = (C_{x_{t-n+1}}, 1 \ldots, C_{x_{t-n+1}}, d, C_{x_{t-n+2}}, 1 \ldots, C_{x_{t-n+2}}, d, \ldots C_{x_{t-1}}, d) \tag{3.25}$$

In general, C maps the sequence of feature vectors of words to conditional probability distribution over all the words in V, and the output is a vector whose element gives an estimation of the probability of the distribution as seen in Figure 3.7.

In training a NNLM to compute the probability of the next word, Bengio et al. [55] proposed using NN architecture with a softmax activation function introduced by Bishop [56]:

$$p(x_n = k|x_{t-n+1} \ldots, x_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}} \tag{3.26}$$

where the unnormailzed log-probabilities are:

$$a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j) \tag{3.27}$$

and the activation function is applied to each element W. The objective is to find parameter θ that maximizes the log likelihood of the training set:

$$L(\theta) = \sum_{t} p(x_t|x_{t-n+1,\ldots x_{t-1}}) \tag{3.28}$$

and parameters b and c are vectors concatenated with parameters W, V. h is the number of hidden units and d is the learned word features. Gradient-based optimization

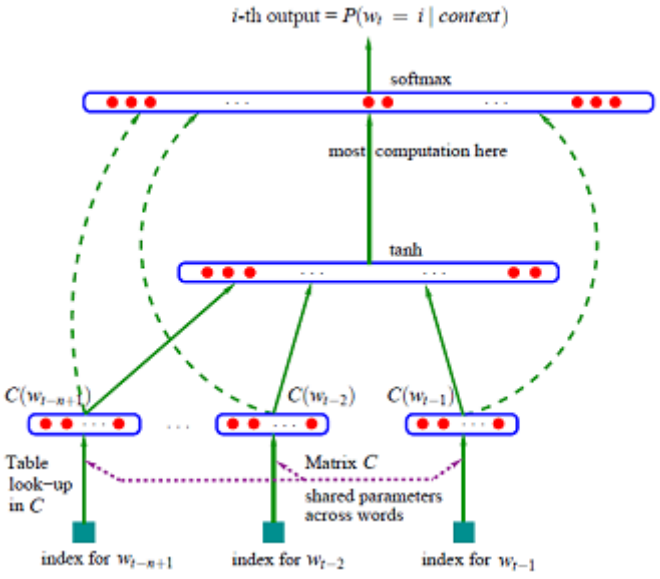

FIGURE 3.7: Architecture of the NNLM introduced by Bengio et al. [55]

algorithm optimized for large number of data such as stochastic gradient descent is used to train the Neural Network architecture in order to maximize the log-likelihood of the training set. The gradient is computed using the error back-propagation algorithm at the projection layer. This gives the idea that NNLM learns the projection of words onto the continuous space with the optimal probability estimation. The only challenge to the NNLM is in its high complexity which is mainly caused by the large size of the output

layer. The complexity is given as:

$$Q = (n-1) \times P \times H + H + H \times N + N \tag{3.29}$$

where P denotes the size of projection, H is the size of the hidden layer and N is the size of the output later. Schwenk [57] proposed the following improvements for using the NNLM:

- Predicting only a subset of the whole vocabulary
- Collecting and sorting the language model probabilities so that same context lead to only one pass.
- Optimizing the CPU performance using machine-specific BLAS libraries for faster matrix operations.
- Using Block mode for faster matrix operations

Schwenk and Gauvain [58] followed the listed methods above to train NNLM on transcriptions of acoustic data(4 million words), commercial transcriptions (88.5 million words), Newspaper texts(508 million word), web data (13.6 million words). They achieved perplexities of 107.4, 137.8, 103 and 136.7 respectively. Interpolating all the corpus (614 million words), they achieved a perplexity of 70.2. The NNLM achitecture used for the acoustic data has a continuous word representation of 50 dimensions, one hidden layer with 500 units and an output layer limited to the 8192 most frequent words. 3.2 million parameters for the continuous representation of words and about 4.2 million parameters for estimating the probabilities were generated. The learning rate of the stochastic back-propagation was set to 0.005 to prevent overshooting the minimum point. He further stated that increasing the number of dimension of the projection layer led to faster convergence and lower perplexity and word error rates.

Chapter 4

Related Work

Over the years, understanding web query intents has proved to be crucial in improving the relevance of search results in search engines. Considerable number of studies and research have been devoted to this, proposing techniques which substantially belong to the Natural Language Processing (NLP) paradigm; such as stop-word removal, stemming, part-of-speech tagging, compound word splitting etc. Applying all these NLP methods to queries and documents, the objective is to fetch the most relevant documents according to user's request which best matches the weighted term representations of documents (indexes). However, all these NLP methods arguably have mixed effects. For example, *"To be or not to be"* will be handled poorly by stop-word removal, stemming will introduce ambiguities for *"organization"* to *"organ"*, part-of-speech tagging: *"Import"(Noun)* versus *"Import"(Verb)*, compound word splitting *"Tell-tale" = "Tell" and "Tale"* etc. Nonetheless, NLP techniques have proved useful for processing queries, especially longer queries. But shorter queries are unable to benefit from NLP. Strzalkowski et al. [59] pointed out that this is because longer queries provide larger amount of contextual information useful for NLP compared to shorter queries which lack enough contextual information that is useful to NLP methods.

Non-NLP related approach to understanding queries are rooted in techniques such as query segmentation, query parsing and query classification. Query Segmentation reduces query into word-based or phrase-based units suitable for inverse look-up (Risvik et al. [60], Tan and Peng [61]). Query parsing deals with extracting linguistic structure of a query in order to find putative phrases in the query (Gao and Nie [62], De Lima and Pedersen [63]). Query classification focuses on classifying query either according to search intent e.g. informational, navigational and transactional (Lee et al. [64], Rose and Levinson [65]) or according to its semantics such as "Sports", "Movies" etc. (Beitzel et al. [66], Shen et al. [67]). Over the last few years, several methods for classifying

queries according to semantics have been proposed. According to reports from the KDDCUP 2005 web search query classification competition (Li et al. [68] and Cao et al. [69]), majority of submitted entries fell into three categories. The first category utilized external information to enrich queries(e.g. Broder et al. [70], Shen et al. [67]), the second category leveraged on labeled data to train machine learning algorithms for predicting query class memberships(e.g. Beitzel et al. [66], Beitzel et al. [71]) and the third category expanded the provided labeled data using click-through to automatically label more queries(e.g. Li et al. [72]). All these techniques were directed towards improving semantics for accurate classification.

To the extent of our knowledge, there have been a few research efforts in attempting Named Entity Recognition techniques for understanding query intents as specified in this thesis. Most recent work on Named Entity Recognition in Query (NERQ) by Guo et al. [73] explored topic models(i.e. Latent Dirichlet Allocation) with a weekly supervised learning method (WS-LDA) to recognize the named entity in query, and then assign the most likely class label to it. For example, given query q, the task of WS-LDA is to find the most likely entity class label represented by the triple (e, t, c) where e is the named entity, t is the context of e in q, and c is the class label of e. However, the WS-LDA in NERQ focused only on queries with single named entity belonging to four classes.

All the query classification and named entity recognition methods focused on classifying a query to a single class, they did not explore the internal structure of the query composition to identifying named entity class of each token in the query. In contrast, we considered the internal structure including the sequential nature of words to infer named entity classes. We also considered queries with multiple named entities.

Chapter 5

Feature Engineering

5.1 Overview

Machine learning has good potentials for helping researchers comprehend classification problem. But the challenges involved is attributed to identifying relevant features useful for the learning task. The performance of a machine learning algorithm ultimately depends on how selected features are able to capture the characteristics of the training examples for the learning outcome. In NER systems, named entities are identified and classified according to underlying features which are characteristics of different linguistic forms of the words in the training examples. Features according to Sekine and Ranchhod [74] can be defined as unique properties or characteristics of texts which contributes information about words, sentences or corpus of documents, useful for algorithmic purposes.McDonald [75] outlined two categories features in NER task can belong to; local and global features. Local features describes characteristic attributes of the sequence of words that makes up an entity whereby global features describes the characteristic attributes derived from the context around the named entities.

5.2 Feature Selection

Training examples are often represented by large number of features out of which only a few are useful for predicting the target labels. Oliveira et al. [76] reported that beyond a certain threshold, the addition of extra features does not only become useless to the performance of a learning model but can even worsen the performance, therefore prompting the need for a procedure to selecting only relevant features from a vast array of options. Feature selection therefore is the process of choosing small but effective

subsets of features from the often redundant large sets of available features, which are relevant to the objective of the learning algorithm .In general, the best fitting feature for any learning algorithm is a subset which mostly contributes to the accuracy of the learning algorithm with the least dimensions, thereby not contributing to the problem of *curse of dimensionality.*

Many feature selection techniques have been suggested in the past, but the current paradigm in feature selection is based on the optimization of multiple objectives, which means balancing trade-offs between minimizing the number of feature and maximizing the quality. Feature candidates which satisfy this condition are sought out from subsets of features according to an evaluation function. Searching for feature candidates from subsets of features this way can be too costly(2^Ncandidates) for any feature set. Most feature selection methods are based on reducing the search computational complexity by using a stop criterion in order to prevent a full-scale search of all the subsets of features. Dash and Liu [77] proposed that the process of feature selection can be done in two stages: the process of generation and the process of evaluation. This procedure is illustrated as seen in Figure 5.1 below:

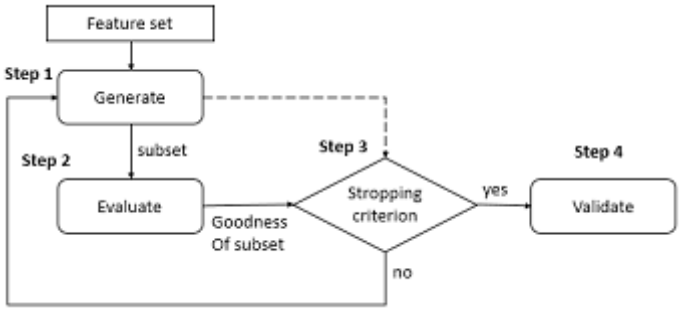

FIGURE 5.1: Feature selection process according to Dash and Liu [77]

The first step in their proposed procedure is the generation process in which subsets of features are being generated either by iteratively adding features or iteratively removing features, depending on how the generation process is started. The possibilities range from starting with no features, or starting with a random subset of features or starting with all the features. The second step is an evaluation process, in which an evaluation function measures how good the current generated subset is compared to the previous one and does the necessary replacement if current subset is better than previous subset or vice versa. The third step acts as the control of the process. It checks the condition of the process against a pre-defined stopping criterion in order to prevent the process from unnecessarily going through all the available feature space. A final evaluation is done in order to validate the result of the selected features.

The generation process earlier introduced by Dash and Liu [77] can be carried out in three different ways, namely:

- complete search
- Heuristic search
- Random search

5.2.0.1 Complete method

This method involves manually creating sets of features to be used in a learning task without any selection of optimal features.(Zhou et al. [78])

5.2.0.2 Heuristic Search

This method relies on heuristics to select the best features. Forward and backward elimination are common examples of heuristic method.

- Forward selection is an iterative feature selection method in which one starts with no features and at every iteration candidates features that mostly minimizes error on the validation sets are added until there is no significant decrease in the error rate.
- Backward selection is also done iteratively but unlike the forward selection technique, one starts with all the available features and then iteratively remove the ones which increases the error on the validation sets thereby minimizing the error rate until no significant decrease in the error rate is observed.

5.2.0.3 Random Search

This method involves the use of algorithms to select the best features in random step or in a probabilistic step. A disadvantage of this approach is that it might not consider correlations and interdependence of features associated with large search space since the algorithm does not search the whole space at once, but rather in steps.

5.3 Feature selection in Named Entity Recognition

The combination of n number of features describing a particular token can be represented as n-dimensional vector during training. Examples of such features in NER systems can be a boolean variable assigned a value *1* for an hyphenated word or assigned a value *0* if otherwise. So by defining features using either local knowledge i.e.local information of the word token and its surrounding context or external knowledge such as gazetteers, part-of-speech tags, chunking etc., a word representation can be constructed and a machine learning algorithm can then learn these rules and use it to generalize over new unseen texts.

5.3.1 Features in NER

- The simplest feature at the level of local feature is string of frequently occuring token converted to lower-case. This lexical feature constitute the majority group of all groups of features since each token is a feature. Infrequent tokens are often discarded during training.
- Capitilzation is also a commonly used feature in NER tasks, for example feature CAPITALIZED is set to 1 if the current word is a capital letter or 0 if otherwise.
- Another common feature is the first word of sentence, a feature First WOrd is set to 1 if current word is the first word or 0 if otherwise. This feature can be combined with feature capitalization to detect proper nouns nouns in a sentence. For example, if current is capitalized and current word is not the first word in a sentence.
- Part-of-Speech tagging defines the syntactic class a word belong to.
- Pattern of digit can also be useful for named entities such as dates, money, percentages etc.
- Dictionaries and Gazetteers refers to collection of words listed in the predefined categories of the named entities. An NER system then do a look-up on the dictionaries in order to find which entity category a word belong. For example, if the word "android" is found in the Operating System dictionary, android is then enriched with the feature "Operating System". Dictionaries and Gazetteers can be used as a feature during training of learning algorithms such that a word found in a typed dictionary will have a higher probability of belonging to the named entity type of the dictionary. Sources of dictionaries can be knowledge bases like DBpedia, FreeBase etc., Wikipedia or a locally curated repository. Lookup techniques

is important to the performance of the system using dictionaries and gazetteers. Common string search techniques implements exact match, in which the occurrence of a pattern of a set of strings forming a word, is located in a target of fixed length. i.e. given a target $T = [1, \ldots, n]$ and a pattern of a set $S = s_1, \ldots, s_m$, an exact match find all the occurrences of S in T. Aho-Corasick Algorithm, on the other hand linearizes the time complexity of string matching by constructing a keyword tree (or trie) for a set of patterns P

Feature selection is non-trivial for identifying and classifying named entities(Sekine and Ranchhod [74]). Selecting the right and relevant feature that clearly discriminate between different named entity classes can be challenging since examples belonging to same class have similar feature values. If the right features for an NER tasks is selected, the computational cost is not expected to be high or over-fit. Many research efforts have focused on selecting the right features. Vail et al. [79] proposed L1 regularization for selecting features in Conditional Random Fields, Tkachenko and Simanovsky [80] combined local features with external knowledge to train a CRF, Oliveira et al. [76] selected features from orthographic parameters describing word level information.

5.4 Machine learning for Feature Engineering

5.4.1 Distributed Representations

Machine learning algorithms for NER tasks estimates joint probabilities between word sequence which can be challenging due to *curse of dimensionality* associated with discrete spaces. For example, modeling a joint probability between N words in a vocabulary size of D leads to D^N free parameters. Distributed representation was initiated from inspiration drawn from earlier research in word representations done by Rumelhart et al. [81], which is based on concepts from cognitive representation of entities, in which an object can be efficiently represented using its features that are categorized into active and inactive. This form of representation helps the brain generalize to unseen objects similar to previously seen examples. Distributed representation in NER can be taken as a form of word representation of words in vector space i.e. representing a word with mathematical object usually a vector in which each dimension value corresponds to a feature. In distributed representation, each dimension represent a latent feature of the word (Turian et al. [49])

Distributed representations maps indexed word in dictionaries to feature vectors in n-dimensional space where the dimensions denotes concepts. In this way the joint probability function of word sequences can be expressed in terms of the feature vectors.

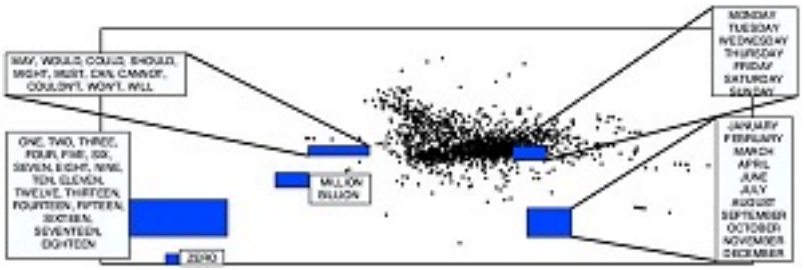

FIGURE 5.2: Bengio [53] 2-dimensional distributed representation of words from Blitzer et al. [82] by Bengio [53]

word	C_1	C_2	C_3
skype	-0.040339	-0.060776	0.070254
whatsapp	-0.062906	0.100627	-0.051664
download	-0.962525	-0.667468	0.605663
install	-0.828293	-0.486217	-0.023461
full	0.340240	-0.101377	-0.055129
free	0.425247	0.971151	-0.136211

TABLE 5.1: Word embeddings

Therefore, a Neural Network Language Model(NNLM) is able to generalize on unseen words because similar words are expected to have similar feature vector and consequently, closer in the n-dimensional space (see Figure 5.2). For example, if it is assumed that "*install*" and "*download*" plays similar semantic role represented with similar feature vectors, and so likewise for the pairs "$full, free$", "$skype, whatsapp$", therefore an NNLM can easily generalize from "download free skype" to "install full whatsapp" and likewise "install full skype" and "download free whatsapp"

The NNLM distributed representation also known as word embeddings was introduced by Bengio [53]. He learned word vectors which can be represented as linear translations. For example, his model calculated the vector of a translation vec("Madrid")-vec("Spain") + vec("France") to a vector which is closest to vec("Paris") than any other word. Other researchers adopted this architecture due to its simplicity and further expanded it. For example, Collobert et al. [83] combined the word embeddings with a convolution architecture to develop a state-of-the-art NLP tool(called SENNA system) for NER, POS tagging, chunking and Semantic Role Labeling. Google used the concepts of word embeddings to develop an image search system by combining word embeddings from queries with image representations in the same space(Weston et al. [84]), Srivastava and Salakhutdinov [85] extended it by using a deeper multi-modal representations. NNLM has been shown to perform better than other known architecture for learning distributed representation of words. Mikolov et al. [86] reported that NNLM performed better than Latent Semantic Analysis (LSA) for preserving linear regularities in words

and Latent Dirichlet Allocation (LDA) leads to expensive computations for large data sets.

5.4.2 Learning distributed representation of words

This section discusses state-of-the art techniques in learning distributed representation of words. As distributed representation using NNLM became more popular, two neural network architectures were more prominent for learning distributed representation. These architectures are Feedforward NNLM and Recurrent NNLM.

5.4.2.1 Feedforward NNLM

In Feedforward NNLM, the input layer projects 1-of-V encoded N previous words to a projection layer P of dimensionality N×D using a shared projection matrix. the projection from the input is not computationally expensive because at any given time only N inputs are active. But computation between the projection layer and the hidden layer becomes complex because of the dense values in the projection layer. Mikolov et al. [87] cited an example where N = 10, the values of the projection layer ranges between 500 to 2000 while the units of the hidden layer ranges between 500 to 1000. The hidden layer computes the probability distribution for all the words in vocabulary, leading to output layer of dimensionality V. The complexity of this architecture per training becomes:

$$Q = N \times D + N \times D \times H + H \times V \tag{5.1}$$

Bengio et al. [88] among others proposed hierarchical softmax solutions which avoids the dominating term H × V, Collobert and Weston [89] proposed avoiding normalized models during training. In the Skip gram model by Mikolov et al. [90], hierarchical softmax which is an activation function in neural network is used to encode the vocabulary in form of a Huffman binary tree. The number of output units needed for evaluation is reduced to $\log_2$(unigram perplexity(V)) by the Huffman trees because it encodes frequent words with binary codes.

5.4.2.2 Recurrent NNLM

The Recurent NNLM on the other hand is able to represent more complex patterns than the feedforward NNLM. The architecture of the Recurrent NNLM has the input layer, the hidden layer and the output layer and no projection layer unlike the feedforward NNLM. It uses time-delayed connections among the connections of the hidden layer

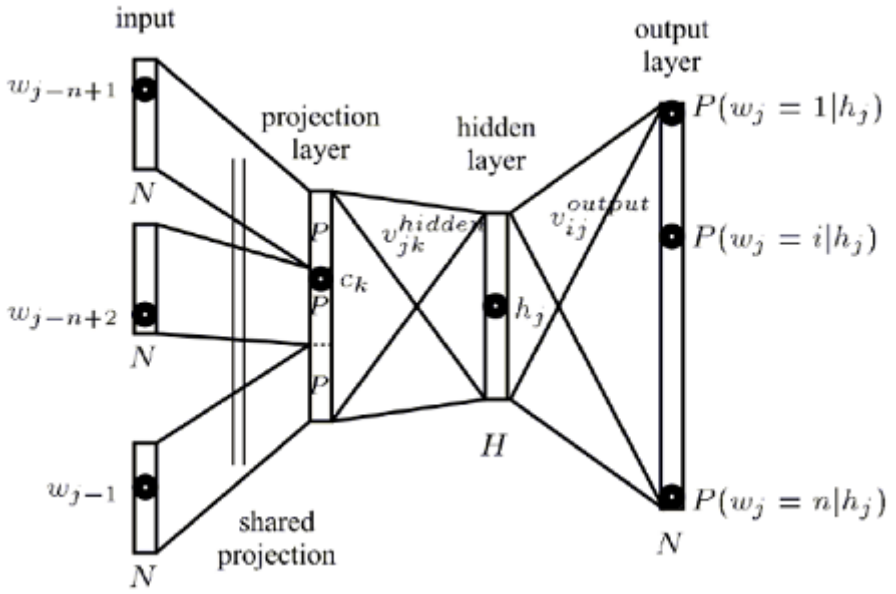

FIGURE 5.3: Architecture of a Feedforward NNLM by Dash and Liu [77]

enabling it for some short-term memory which can store information from the past and get updated based on the current input and state of the hidden layer in previous time step. The complexity of the Recurrent NNLM architecture per training is:

$$Q = H \times H + H \times V \tag{5.2}$$

where dimensionality of the hidden layer is the same as the dimensionality of the word representation. Just like in feedforward NNLM, hierarchical softmax can also be used to the reduce the complexity H × V to H $\times log_2 V$.

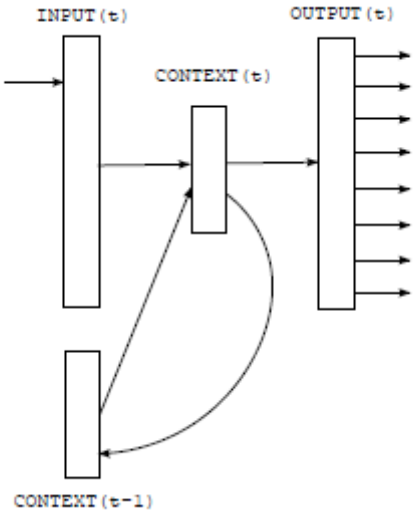

FIGURE 5.4: Architecture of the Recurrent NNLM Model introduced by Mikolov et al. [87]

Mikolov et al. [90] further improve on the architecture of feedforward NNLM by proposing two new architectures which have less complexities compared to feedforward NNLM and Recurrent NNLM. They observed that the non-linear hidden layer in both previous architectures contributed majorly to the complexities, and therefore suggested an N-gram approach which is more simple and efficient. Their architecture involves learning a continuous word vector using a simple model and then training an N-gram model NNLM on it.

5.4.2.3 Continuous Bag-of-Word Model(CBoW)

This is somewhat similar to the feedforward NNLM in that its architecture also has a projection layer, but it is explicitly different from the feedforward NNLM because it does not have a hidden layer and also because the projection layer is shared by all words as illustrated in Figure 5.5 i.e. all words are projected into the same position: both past and present contexts, $w(t-2), w(t-1) \ldots, w(t+2)$. Since the projection layer is a Bag-of-Word, position or order of word in the history does not matter. So for a given word, a log-linear classifier was used to compute the probability of that word given the past and present context. The training complexity for this architecture is

$$Q = N \times D + D \times \log_2 V \tag{5.3}$$

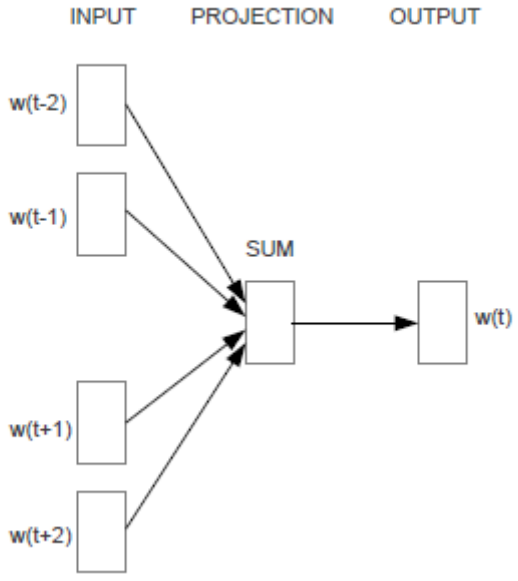

FIGURE 5.5: Architecture of the Continuous Bag-of-Word Model introduced by Mikolov et al. [90]

5.4.2.4 Continuous Skip-gram Model

In the architecture of the Continuous Skip-gram model, each word is an input to the log-classifier with a continuous projection layer as illustrated in Figure 5.6. It computes the probability of a context given a word i.e. predict words within a range of words by maximizing the probability of the context, given that word. Increasing the size of the context(increase in complexity) increases the quality of the prediction. The training complexity of this architecture is

$$Q = C \times (D + D \times \log_2 V) \tag{5.4}$$

where C denotes the maximum distance of words, D is the word representations and V is the size of the vocabulary

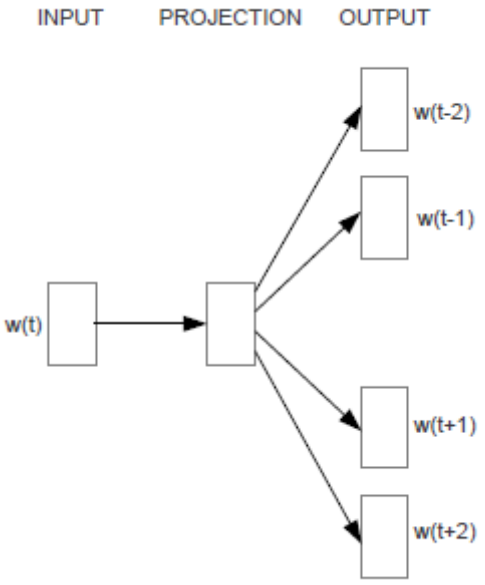

FIGURE 5.6: Architecture of the Skipgram Model introduced by Mikolov et al. [90]

The training objective of the continuous skip-gram model can be expressed according to Goldberg and Levy [91] as follows;

Given a set of all words D, a word w, and its context c. The objective of the continuous skip-gram model is to obtain the parameter θ that maximizes the probability of the context given the word:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta) \tag{5.5}$$

The basic Skip-gram can be defined using softmax function:

$$p(c|w;\theta) = \frac{e^{(v_c)\cdot(v_w)}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}} \tag{5.6}$$

where v_c and $v_w \in R^d$ are vector representations for contexts and word, and C is the set of all available contexts. Computing $p(c|w;\theta)$ in this way is too expensive because of $\sum_{c' \in C} e^{v_{c'} \cdot v_w}$ over all the context c' which is proportional to the number of words in the vocabulary, Mikolov et al. [92] proposed an efficient approximation of softmax called hierarchical softmax to be used. The hierarchical softmax represents the output layer as a binary tree having all the words as the leaves and the probabilities of child nodes i.e. each word can be reached by a path. In this way, $p(c|w;\theta)$ is now proportional to length of the path to reach word.

Chapter 6

Experiments and Results

In this chapter, we will be outlining the different experiments we carried out for this thesis work with full details of the experiment setting and the procedures we followed. Firstly, we will report details drawn from exploring the data sets for this project, then we will analyze the characteristics in order to understand the data and how to design the methods for the experiments.

Secondly, we will put together resources in terms of tools and software packages for conducting our experiments. We will prepare our training data carefully supervised by domain experts and label them for training purposes. In this experiment, we will examine the impact of the size of training data in machine learning experiments by conducting an initial named entity recognition for a series of training instances. This we assume will give us hints about what to expect considering the limited human supervision during the preparation of the training data sets.

Thirdly, we will experiment using the selected algorithms and report findings. We will discuss reasons for our results and propose how the performance can be improved when necessary.

6.1 Task and Dataset

The dataset for our experiment was collected from search logs stored in Hadoop clusters at Softonic. Our dataset consists of top 1000 queries issued by users from the English web search portal. We compiled this list of queries as Hadoop jobs sent to retrieve data from the clusters, and then save the results as a single file. The data file consists of two columns, the search text and the instance counts of these texts. Graphically exploring our data file as seen in Figure 6.1, shows that the search text varies in length. We

observed that 60% of the queries are of word length between 2 to 8. Specifically, query texts with two word-length dominates the whole distribution. Each one word query (e.g. *skype*) or group of words query (e.g. *microsoft word*) forms the word categories to be identified. There is a variation in word combinations that makes up labels in our data, and we lack enough context in the queries as discussed in our problem statement 1.3.

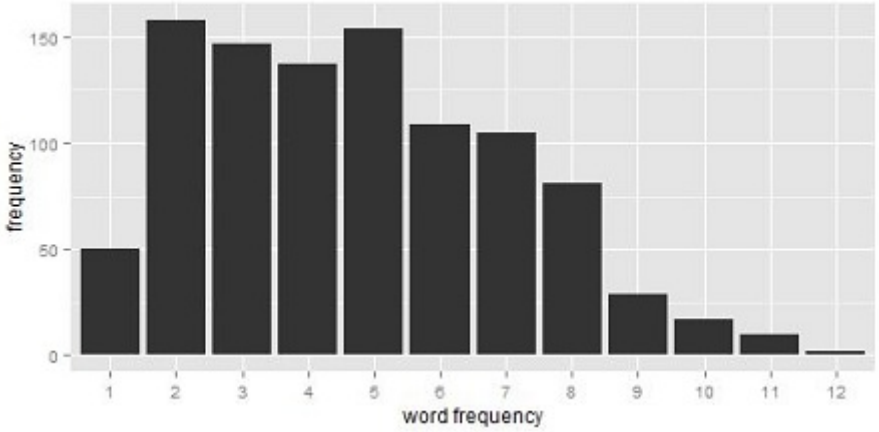

FIGURE 6.1: Frequency of word length in search query

Each line in our dataset is an instance of query, which we will pre-process and annotate accordingly. Our taxonomy of pre-defined named entity classes for this task is defined as follows:

1. **Program**, refers to the named programs. Examples include *ares, skype, microsoft office etc.*
2. **Version** refers to different versions of named programs.
3. **Licence** refers to different types of licences for named programs. Examples are *free, full, premium, trial etc.*
4. **Category** refers to generic descriptions used in place of named programs. For example *game, audio, mp3, movie etc.*
5. **Operating System** refers to different operating systems named programs runs on. Examples include *android, windows, ios etc.*
6. **Device** refers to physical devices that runs named programs such as *tablet, ipad, phone, nokia*
7. **Action** refers to user actions expressed in the queries. Examples are *download, install etc.*

6.1.1 Data Annotation

We manually annotated our dataset using domain knowledge. Each word or group of words which from domain knowledge we identified as named entities were labeled according to memberships in the pre-defined classes. For this use case, we assumed named entities do not overlap and are non-recursive. After annotating our dataset, we explored its characteristics to gather information regarding the distribution of named entities in our data.

Figure 6.2 shows the distribution of named entities in our dataset, and we observed that around 80% of all entities present in our data set belong to the class program name and license.

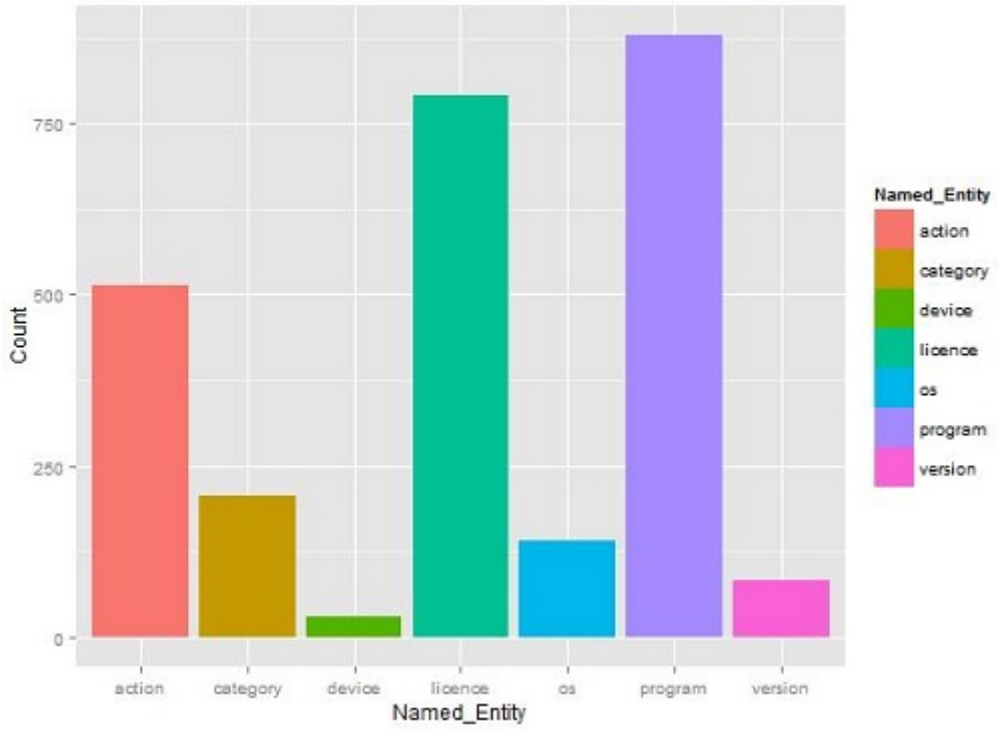

FIGURE 6.2: Distribution of Named Entities after annotating our data

6.1.2 Experimental Settings

We developed a python framework in IPython Notebook for the purpose of our experiments in order to leverage on the parallel computing abilities of IPython and the possibility to include inline documentation while writing the codes. Our infrastructure consist of a computer with the following configuration:

- Processor: Intel(R) i5 2.5GHz Quad core CPU

- Memory: 8 GB
- System type: 64-bit Operating System
- Operating System: Windows 7

6.1.3 Algorithms

We conducted our experiments following two NER approach: We implemented the dictionary-based NER and the machine learning NER(supervised learning). We planned to exploit results from our dictionary-based approach to improve our machine learning approach to NER. We selected HMM, CRF and Neural Network for our supervised machine learning, mostly because of reported performances in earlier NER tasks. Our experiments thus followed the steps listed below:

1. Dictionary Named Entity Recognition
   (a) Dictionary Construction
   (b) Dictionary look-up
   (c) Evaluation
2. Machine learning based Named Entity Recognition(Supervised Learning)
   (a) Hidden Markov Model
      - Train HMM using labeled examples
      - Test the model and report evaluation
   (b) Conditional Random Field
      - Train CRF model using labeled examples
      - Test the model and report evaluation
      - Generate more features and re-evaluate
   (c) Neural Network
      - Generate word embeddings from unlabeled data using Skip-gram Model
      - Train a Multi-layer Neural Network using word embeddings as input
      - Evaluate the performance

6.1.4 Evaluation Method and Result Analysis

The evaluation of the performance of the algorithms are based on standard evaluation metrics in machine learning. We made an assumption in our case that named entities in domain-specific queries do not overlap unlike in general search engine queries. This is because domain-specific queries are not in natural language. Therefore, we defined the evaluation metrics over each token in the training data as follows:

1. Precision, percentage of predicted named entities that were correct:

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

2. Recall, percentage of named entities and non-named we were able to predict

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

3. F-Score, harmonic mean of precision and recall

$$F - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6.3}$$

where
TP (True Positive): correctly predicted named entities
TN (True Negative): correctly predicted non-named entities
FP (False Positive): wrongly predicted named entities
FN (False Negative): wrongly predicted non-named entities

6.1.5 Issues and Challenges in the Experiments

It is generally believed that machine learning algorithms perform better with increase in labeled examples. In our case, we did not have enough human resources to label large texts of corpus. So we restricted our experiments to the available dataset we were able to manually annotate ourselves. We trained and tested using a sequential tagger model (i.e. CRF, HMM), using the manually annotated data, and then evaluated the performance of the model using k-fold cross validation technique. For this technique, we divided our data into K equal parts as illustrated in Figure 6.3, and then set the size of the training set to 80% and test set to 20% of the dataset. We then repeated the hold out training and testing K times, and for each iteration we select one of the K subset as the test set and K-1 subsets as the training set. We then computed an average error on

all the K trials.

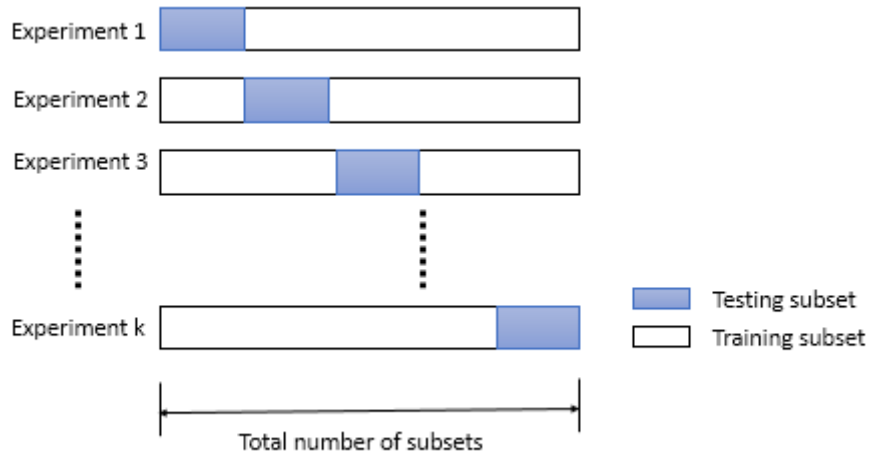

FIGURE 6.3: Illustration of k-fold cross validation

Figure 6.4 shows the result of the performance with increasing datasets. We observed that performance improves as the number of training examples increases.

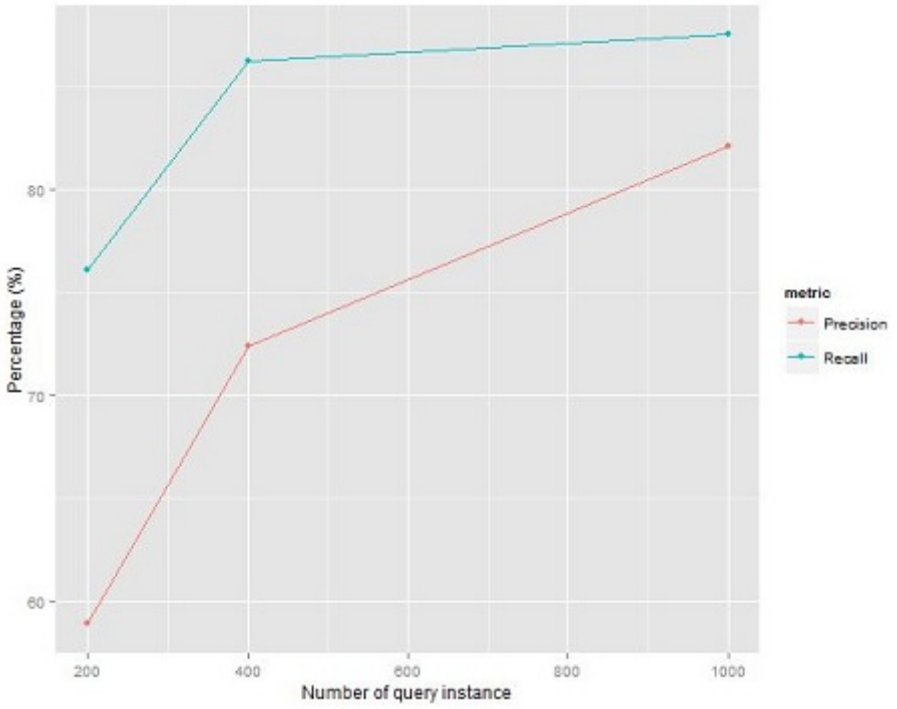

FIGURE 6.4: Illustration of performance with increasing data.

6.2 Dictionary-based Named Entity Recognition

Our dictionary-based Named Entity Recognition system was implemented in Scala, an object-functional programming language. Our dictionary string matching was based on Aho-Corasick algorithm. The architecture of our dictionary-based NER is depicted in the Figure 6.5, consisting of raw queries which are the input, an implementation of Aho-Corasick algorithm for string matching, test set, outputs and an evaluation process. Our dictionaries are ordered in such a way that each word in the query sequence is looked up in the ordered dictionary one after the other, and if a match is found, we move to the next word in the sequence. This was done in order to prevent multi-classification of words that exists in more than one dictionary. For example, the word *"download"* can be found in action dictionary as *"download"* and equally in program dictionary as *"internet download manager"*.

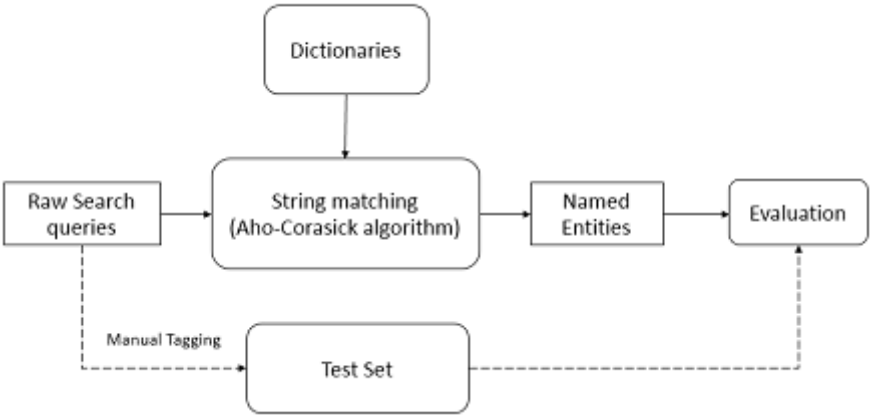

FIGURE 6.5: Architecture of our Dictionary-based NER

6.2.1 Dictionary construction

We constructed named entity dictionaries of about 120 995 entries compiled from data repositories at Softonic. The repository hosts detailed information about available programs that could be searched from the program download portal. The information spans from full program names to manually curated descriptions by individual Program Uploaders and Content Editors at Softonic. This makes the repository a suitable source for constructing a dictionary. We created dictionaries for each target classes such as program names, versions, licenses, actions, categories, operating systems and devices. Table 6.1 depicts sample information in our dictionaries:

Named Entity Dictionaries	
PROGRAM	Microsoft Download Manager Microsoft Excel 2011 Microsoft Expression Media ...
VERSION	Microsoft Download Manager 1.1 Microsoft Excel 2011 14.0.0 Microsoft Expression Media 1.0 ...
LICENSE	free full ...
ACTION	download install ...
...	...

TABLE 6.1: Candidates named entities in the Dictionaries

6.2.2 Dictionary look-up and String Matching

Named entity dictionaries are constructed from the program repositories containing all the different named entities to be identified according to the classes they belong. Then, a string matching system which receives instances of search queries, matches the query instances against the dictonaries. Matched words in each instance are then tagged according to the class of dictionary where a match is found. The string matching system is an implementation of the Aho-Corasick algorithm by Aho and Corasick [93], which is a kind of dictionary matching algorithm that searches texts from a finite set using pattern matching machine in a single pass. According to the illustration in Figure 6.6, a tree is constructed for what is to be searched, where nodes in each level of the tree contains characters. The algorithm which builds up a finite state machine searches by checking the first character of the input against each node and thus outputs nodes where matches are found into a list and subsequently searches the next characters in this manner until it reaches the terminal node. In this way, the algorithm is able keep information(the list) about nodes for every possible match.

6.2.3 Result and Discussion

We approached the evaluation methodology for the dictionary-based from the human linguist point of view, considering only the fully matched queries. We defined fully matched queries as queries in which all words in the queries were full matched to an entry in the dictionaries. We evaluated the recall of this approach, calculating how much relevant queries were retrieved using this approach. From Figure 6.2, we can see that

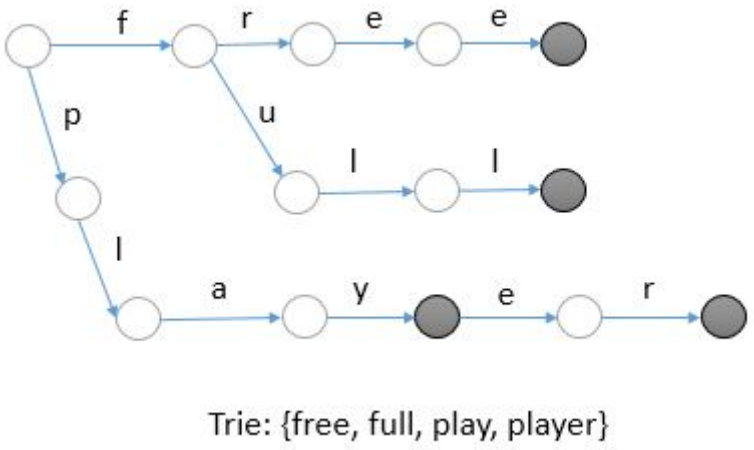

FIGURE 6.6: Aho-Corasick string matching

Queries	Dictionary (entries)	Full Match ratio(%)
1 000	120 995	19.8

TABLE 6.2: Evaluation result for Dictionary-based NER

the Dictionary-based NER has a low recall of about 19.8%. This means that in 1000 queries, only 198 queries were fully matched. And since we are considering full matched queries, our precision is 100%.

This result can be further improved by increasing the size of the dictionary i.e. adding more entries. But in practice, we consider using the dictionary-based as an extra source of feature i.e. gazetteer, for the machine learning techniques such that for each token in the training data, we set the value of a feature to 1 if token is found in the gazetteer and 0 if otherwise.

6.3 Machine learning approach to Named Entity Recognition

For our experiments in machine learning learning-based NER, we followed the supervised learning method. We used HMM and CRF implementation from a Java library called MALLET(McCallum [94]). MALLET is a Java-based package for statistical natural language processing, clustering, document classification, topic modeling and other machine learning algorithms. MALLET's sequential algorithm learning optimization is based on the Limited Memory BFGS (L-BFGS).

MALLET source codes were obtained from the homepage[1] and compiled using the built-in build.xml file. We then prepared our data in the required format suitable for MALLET to be used. For this task, we used our specifically adapted parser and tokenizer to parse and tokenize the queries.

Our parser takes an instance of each query, splits it into word and label, and then tokenizes each word-label pair, using blank space to signify the boundary of each query line. We followed the format in IOB encoding(Zhang et al. [10]) to represent boundaries of each named entity. The IOB encoding is a way of prefixing each word of an entity with a B if the word is a beginning of a named entity, or an I if it is a continuation or inside of a named entity and O if the word is outside any of the named entity.

For example, consider the annotated queries:

```
<prog>skype toolbar</prog><lic>free</lic>version
<ac>download</ac><prog>ares</prog><lic>full</lic>version
```

After tokenizing and encoding, they become:

```
skype B-PROG
toolbar I-PROG
free B-LIC
version O

download B-AC
ares B-PROG
free B-LIC
version O
```

6.3.1 Hidden Markov Model

Our experiment with HMM is illustrated in Figure 6.7. The expected outcome is a model trained using HMM implementation of MALLET. We splitted our data sets into train and test sets, with 80% as training set and 20% test set, we trained HMM with the training set and evaluated the performance of the output model on the test set. Each instance of our training set contains one or more named entity labels from the following components:

[1] http://mallet.cs.umass.edu/index.php

Model	Precision(%)	Recall(%)	F-score(%)
HMM	82.84	82.07	82.45

TABLE 6.3: Evaluation result for HMM

- entity class PROG, VER, LIC, OS, AC, CAT, DEV
- entity boundary B-Beginning, I-Inside, O-Outside

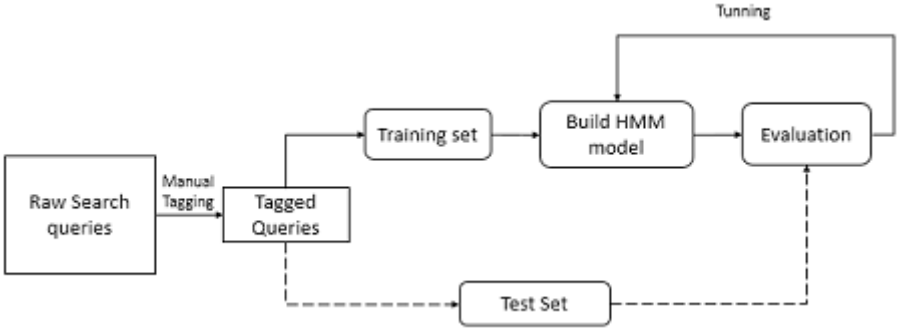

FIGURE 6.7: HMM Architecture for NER

6.3.1.1 Result and Discussion

In the following, we present result of our experiments from 698 features that were generated in MALLET. From Table 6.3, we can see that HMM achieved a precision of about 83% and a recall of about 82%. We believe that increase in the number of training examples would improve the performance of the model created from HMM

6.3.2 Conditional Random Field

For our CRF experiment, we trained CRF model using MALLET's implementation of CRF algorithm on 80% of the corpus, and we used the remaining 20% to test the model. Each instance of the train and test set contain one or more named entities belong from the following:

- entity class PROG, VER, LIC, OS, AC, CAT, DEV
- entity boundary B-Beginning, I-Inside, O-Out of boundary

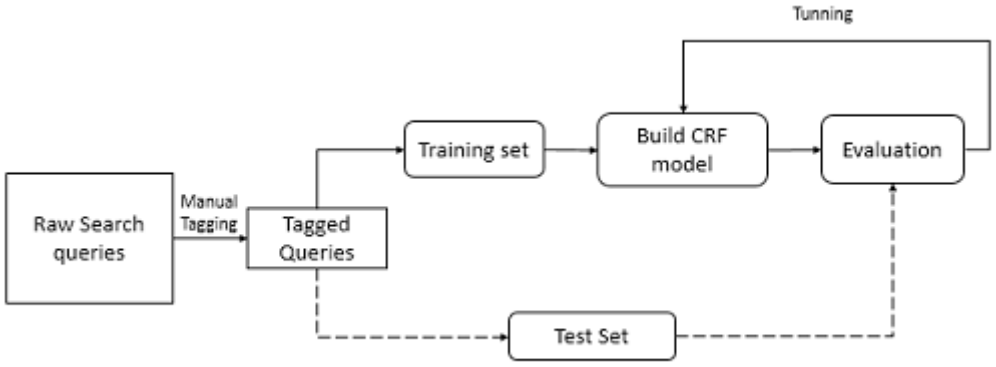

FIGURE 6.8: CRF Architecture for NER

Model	Precision(%)	Recall(%)	F-score(%)
CRF	85.60	86.66	86.13

TABLE 6.4: Evaluation result for CRF model

6.3.2.1 Result and Discussion

In the following, we present Table 6.4, result of our CRF experiments from 698 features generated in MALLET and initial weights of 0. The precision was 85.60% with a recall of 86.66%. The CRF model without any additional feature performs better than HMM. In the next section, we will be focusing on generating more features and measuring the performance. Theoretically, CRF being a discriminative model benefits from additional features such as morphology, capitalization, gazetteer etc. HMM being a generative model, does not.

6.3.3 Feature Engineering

In the next phase of our CRF experiment, we trained our CRF model with more features by introducing morphological and gazetteer features into our training data and reporting observation in the performance of the model

Morphological feature

We generated a binary morphological feature which returns true or false if token matches a regular expression pattern for program version:

```
(?:(\d+\.(?:\d+\.)*\d+))
[x.xx, x.x.xxx, x.x.x.xxx.. ]
```

Gazetteer

A gazetteer contain a list or dictionary of candidate named entities we desire to find. Gazetteer provides extra features in CRF by modeling the dependencies of a label and gazetteer class. Search operations of query tokens are performed on each named entity class of the gazetteer in order to find matches, and also return the entity class of the matching gazetteer. We used the named entity dictionaries as gazetteers, in which a binary feature is set to true if a match is found or false if no match. We also extended this by return the named entity class of the matching gazetteer.

In MALLET we do not need to explicitly state the absence of a feature. We generate only the available features that represents the token. The feature representation are generated by encoding the tokens and the features. For example, using bag of words to encode tokens:

$$[\text{"}skype\text{"}, \text{"}download\text{"}, \text{"}5.2\text{""}free\text{"}, \dots]$$

and similarly the features are also encoded as follows:

Token in gazetteer: We do a lookup in the gazetteer for the token and output $True(1)$ if matched or $False(0)$ if match not found

Gazetteer class: $[PROG, VER, LIC, AC, DEV, CAT, OS]$

Token Match version regex pattern: $True(1)$ or $False(0)$

For a labeled training example:

```
skype B-PROG
5.2  B-VER
free B-LIC
download B-AC
```

where vectors are: $[programs][gazetteer][gazetteerClass][matchRegex]$, the training example is represented as:

$$x(\text{"}skype\text{"}) = \Big[[1,0,0,0\dots][1][1,0,0,\dots]\Big],\ y = B-PROG$$

$$x(\text{"}5.2\text{"}) = \Big[[0,0,1,0\dots][1][0,1,0,\dots][1]\Big],\ y = B-VER$$

$$x(\text{"}version\text{"}) = \Big[[0,0,0,1\dots][1][0,0,1,\dots]\Big],\ y = B-LIC$$

$$x(\text{"}download\text{"}) = \Big[[0,1,0,0\dots][1][0,0,0,1\dots]\Big],\ y = B-AC$$

Model	Precision(%)	Recall(%)	F-score(%)
CRF	87.40	88.26	87.83

TABLE 6.5: Evaluation result for CRF model after generating more features

6.3.3.1 Result and Discussion

The result of the evaluation can be seen in Table 6.5. It shows improvement of about 2% in the result due to the extra features gained from the gazetteer and the regex pattern match for finding version named entities. The new precision is now about 87% with a recall of about 88%.

6.3.4 Neural Network

Our Neural Network experiment was conceived and carried out in two phases. The first phase aimed at exploring the use the of NNLM technique to induce distributed representations of word embeddings which will be plugged into a Neural Network algorithm. The second phase involves the use of a Multilayer Neural Network as a contextual classifier for our named entities. A Multilayer Neural Network receives contextual information by means of a sliding window mechanism which moves across word embeddings of each query instance.

6.3.5 Distributed Representations of search queries

We induced distributed representations of word embeddings of our raw unlabeled corpus(unlabeled version of the same 1000 queries we used for CRF and HMM) using the open source tool called word2vec[2], a deep learning package based on implementation of Skipgram model as proposed by Mikolov et al. [90]. Word2vec is an implementation of the skip-gram model which we discussed in section 5.4.2.4, for computing vector representation of words. In our framework we developed a wrapper for word2vec and used it to extract distributed representations of words in our data. The output from wordvec is a file consisting of each words and its feature vector according to the size we specified. We used feature vector of size 5 (5 dimensions) in order to keep the complexity minimal. However, it should be noted that larger dimensions produces more richer semantics in the distributed representations but requires more time. Figure 6.10 shows our word embeddings on a 2-dimensional representation using t-Distributed Stochastic Neighbor Embedding (t-SNE), a dimensionality reduction package by Van der Maaten and Hinton

[2]https://code.google.com/p/word2vec/

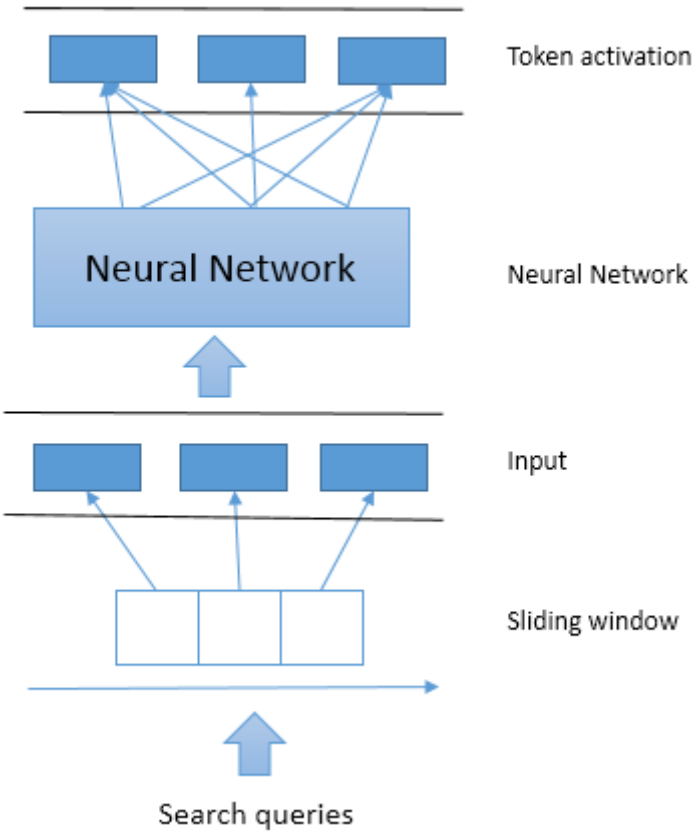

FIGURE 6.9: Architecture of our Neural Sliding window

[95], which is useful for visualizing high-dimensional datasets. From the figure we can see that semantically related words appear together.

Given the vocabulary of words with feature vectors, we followed the steps outlined below to train a Multilayer Neural Network algorithm using 80% labeled queries as our train and 20% queries as test set. Our Neural network receives input by means of window sliding mechanism illustrated in Figure 6.9

Step 1: For each query instance $q \in Q$ split q into list of tokens $T(q) = t_1, t_2, \ldots, t_n$

Step 2: For each token t_i extract feature vector $\vec{v}(t_i) = (f_1, f_2, \ldots, f_m)$ from vocabulary V, concatenate feature vector of $q_i(t)$ tokens

Step 3: Starting at first token, move sliding window W_d of dimension d = 3 across list features, with steps s = 1

Step 4: For each position i of sliding window $W_{d=3}$ use features (f_{t+1}, f_{t+2}) as input to neural network and feature f_t as target label

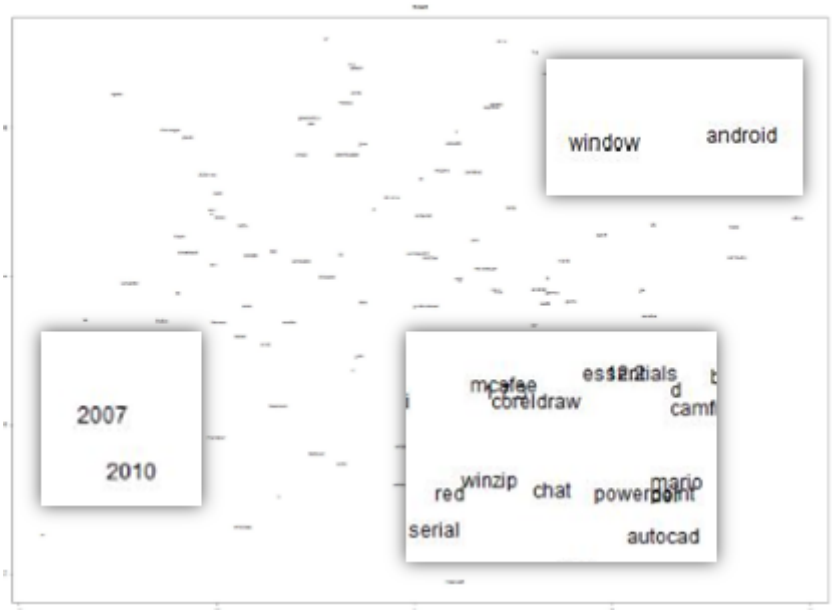

FIGURE 6.10: Zoom-in on the 2-D distributed representation of words from our corpus

Step 5: Compute output, compare with actual, back-propagate error, adjust weights of network.

Step 6: Do until network converge to optimal

We used the neuralnet R package by Fritsch et al. [96] to train and test our Neural Network Language model. Below are the details of our configuration in the *neuralnet* package:

Input: 15 neurons (5 concatenated word embeddings for words in window size 3)

C_1	C_2	C_3	C_4	C_5
-0.486217	-0.040339	-0.060776	...	...
-0.055129	-0.062906	0.100627	...	...
-0.136211	-0.962525	-0.667468	...	...

TABLE 6.6: Word embedding inputs from sliding window(size 3)

Hidden Layers: 25

Activation function: We selected the hyperbolic tangent as our activation function because it is fully symmetric compared to the sigmoid function and due to the ease of obtaining the derivative i.e.

$$x = F(y) = \tanh y, \frac{dF}{dy} = \frac{\cosh^2 y - \sinh^2 y}{\cosh^2 y} \tag{6.4}$$

$$\frac{dF}{dy} = 1 - x^2 \tag{6.5}$$

We selected an intermediate value $\left[+0.9, \text{-}0.9\right]$ similar to Figure 6.11

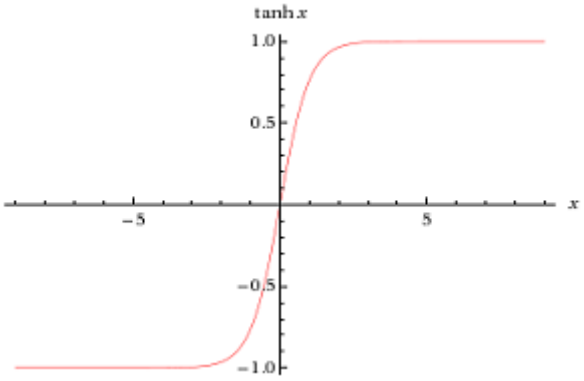

FIGURE 6.11: Hyperbolic tangent activation function

Output: 7 neurons (classes of named entities)

y_1	y_2	...	y_7
-0.9	+0.9	...	...

TABLE 6.7: Output label encoding for Neural Network NER

So considering an instance of our manually labeled training example:

```
skype PROG
free LIC
download AC
windows OS
```

We encoded it using the induced word embeddings, where the output y is encoded as $\left[PROG, VER, LIC, AC, DEV, CAT, OS\right]$, and following the sliding window technique to provide context, the training example is thus represented as:

$x(\text{"}free\text{"}, \text{"}download\text{"})$, $y(skype)$:

Model	Precision(%)	Recall(%)	F-score(%)
Neural Network	93.00	93.00	91.00

TABLE 6.8: Evaluation result for the Neural Network NER

$$\Big[[0.425247, 0.971151, -0.136211\ldots]\,[-0.962525, -0.667468, 0.605663\ldots]\Big] = [+0.9, -0.9, -0.9, \ldots,]$$

$x(\text{"}download\text{"}, \text{"}windows\text{"})$, $y(free)$:

$$\Big[[-0.962525, -0.667468, 0.605663\ldots]\,[-0.132612, 0.123532, -0.028780\ldots]\Big] = [-0.9, -0.9, +0.9, \ldots,]$$

$x(\text{"}windows\text{"})$, $y(download)$:

$$\Big[[-0.962525, -0.667468, 0.605663\ldots]\Big] = [-0.9, -0.9, -0.9, +0.9\ldots,]$$

6.3.6 Result and Discussion

Table 6.8 shows the performance of the Neural Network NER trained on 800 manually labeled instances of queries, and tested with 200 never-seen-before queries. Our Neural Network NER predicted new labels at a precision of 93% and a recall of 93%. Our Neural Network Language model induced word embeddings proved efficient in generating high quality feature vectors that covers semantic and syntactic patterns between words and their context which is needed to overcoming the challenges of word ambiguities. Word tokens with similarities in terms of semantics and syntax were observed to be close to each other in the n-dimensional projection space. i.e. "*window*" and "*android*" (see Figure 6.10)

	PROG	VER	LIC	OS	DEV	AC	CAT
PROG	432	2	0	0	0	1	1
VER	18	3	0	0	0	0	0
LIC	1	0	186	0	0	0	0
OS	13	0	0	9	0	0	0
DEV	9	0	0	0	1	0	0
AC	9	0	0	0	0	120	0
CAT	7	0	0	0	0	0	32

FIGURE 6.12: Neural Network NER Confusion Matrix

We also constructed a confusion matrix as seen in Table 6.12 in order to visualize the performance of our Neural Network model. The row in the confusion matrix represents the true class while column represents the prediction of of our Neural Network. Reading the table diagonally, it can be seen that PROGRAM named entities dominated and Neural Network was able to correctly predict 432 tokens out of 489 tokens, and similarly it was able to correctly predict all the 186 tokens of LICENCE and 120 tokens out of 121 tokens for ACTION.

Chapter 7

Conclusions and Future work

Our objective in this thesis work was to apply machine learning algorithms to recognize named entities in a domain-specific search engine. Initially, we conducted an experiment using the dictionary-based approach for identifying members of our pre-defined named entities from sample queries. We further explored how the dictionary approach can be used to improve the machine learning methods. Our machine learning based approach consisted of three algorithms; Hidden Markov Model, Conditional Random Field and Neural Network. Our dictionary-based approach was used as a basis for a gazetteer in the machine learning algorithms, and we observed that Hidden Markov Model which is a generative model could not benefit from the external features we gained from the gazetteers. Conditional Random Field which is discriminative, on the other hand showed an improvement of about 2% from the extra features gained from the gazetteers and other morphological aspects of the queries, such as the format of program version. These features proved to be essential in recognizing named entities in queries.

The Neural Network NER system was implemented by coupling an induced Neural Network Language model word embeddings with a multilayer neural network in order to predict NE labels. The model followed after the Skip-gram model, emitting multi-dimensional distributed representations of our query. We trained a multilayer Neural Network using feature vectors from distributed representations through a sliding window of three tokens.

Figure 7.1 shows the performance of all the three methods we implemented according to the standard evaluation criteria; Precision, Recall and F-score(F1). Neural Network had 93% precision, a recall of 93% and F-Score of 92%. CRF achieved 87.40% precision, 88.26% recall and F-Score of 87.83%. HMM had 82.84% precision, a recall of 82.07% and F-Score of 82.45%. From this results, we are able to conclude that Neural Network was the best model for this task, it performed better than both CRF and HMM.

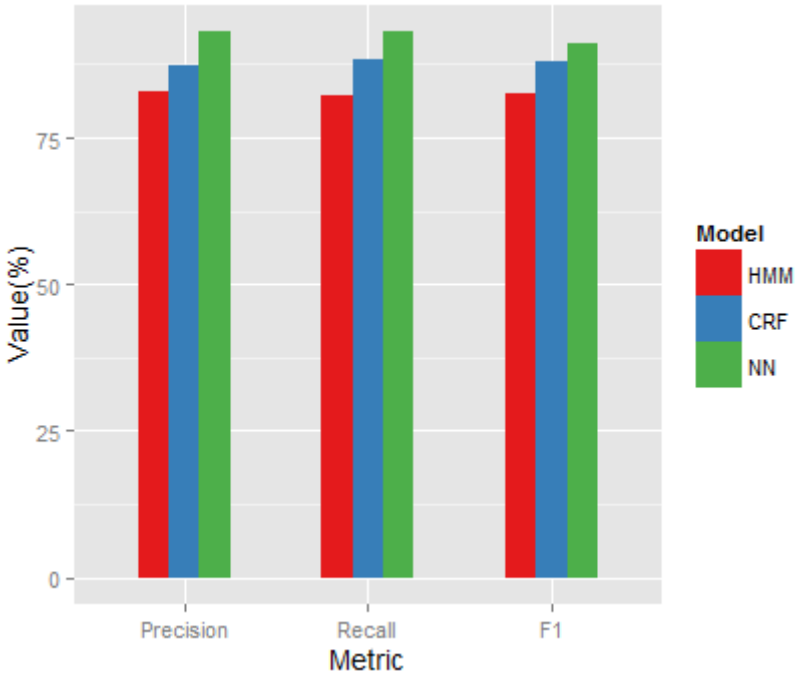

FIGURE 7.1: Evaluation for HMM, CRF and NN

Neural Network Language Modeling using the Skip-gram model were found to be useful for inducing distributed representations, as it provides a linear translation over the whole word tokens. In future, we would like to investigate the effect of enriching the distributed representation with news articles and external knowledge base in order to provide richer context for the training examples. This we believe should have little effect on the complexity since heavy computations are between the hidden and the output layer.

We would also like to explore semi-supervised learning techniques for NER in order to overcome the challenges attributed to manual labelling of training examples in supervised learning. With semi-supervised learning, word embeddings of a small labeled subsets could be used to derive more examples occurring in similar context as projected onto the continuous space. Crowd-sourcing the manual labeling task is also important to us, therefore we would like to investigate how we could make users annotate queries while they search for programs on the Softonic download portal.

Bibliography

[1] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[2] Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David Grossman, David D. Lewis, Abdur Chowdhury, and Aleksandr Kolcz. Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 581–582, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: 10.1145/1076034.1076138. URL http://doi.acm.org/10.1145/1076034.1076138.

[3] Marius Paşca. Organizing and searching the world wide web of facts – step two: Harnessing the wisdom of the crowds. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 101–110, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242587. URL http://doi.acm.org/10.1145/1242572.1242587.

[4] R. Grishman and B. Sundheim. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June 1996.

[5] Shipra Dingare, Jenny Finkel, Malvina Nissim, Christopher Manning, and Claire Grover. A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. in the 2004 biolink meeting at ismb. 6:1–2, 2005.

[6] NV Sobhana, Pabitra Mitra, and SK Ghosh. Conditional random field based named entity recognition in geological text. *International Journal of Computer Applications*, 1(3):143–147, 2010.

[7] Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. Shallow parsing and text chunking: a view on underspecification in syntax. *Cognitive science research paper-university of Sussex CSRP*, pages 35–44, 1996.

[8] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

[9] Antonio Molina, Ferran Pla, James Hammerton, Miles Osborne, Susan Armstrong, and Walter Daelemans. Shallow parsing using specialized hmms. *Journal of Machine Learning Research*, 2:595–613, 2002.

[10] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. *J. Mach. Learn. Res.*, 2:615–637, March 2002. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944790.944820.

[11] Michele Banko, Oren Etzioni, and Turing Center. The tradeoffs between open and traditional relation extraction. In *ACL*, volume 8, pages 28–36. Citeseer, 2008.

[12] Zhenzhen Kou, William W Cohen, and Robert F Murphy. High-recall protein entity recognition using a dictionary. *Bioinformatics*, 21(suppl 1):i266–i273, 2005.

[13] Dimitra Farmakiotou, Vangelis Karkaletsis, John Koutsias, George Sigletos, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Rule-based named entity recognition for greek financial texts. In *In Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000*, pages 75–78, 2000.

[14] Sherief Abdallah, Khaled Shaalan, and Muhammad Shoaib. Integrating rule-based system with classification for arabic named entity recognition. In *Computational Linguistics and Intelligent Text Processing*, pages 311–322. Springer, 2012.

[15] Karthik Gali, Harshit Surana, Ashwini Vaidya, Praneeth Shishtla, and Dipti Misra Sharma. Aggregating machine learning and rule based heuristics for named entity recognition. In *IJCNLP*, pages 25–32, 2008.

[16] Jun'ichi Kazama and Kentaro Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707, 2007. URL http://www.aclweb.org/anthology-new/D/D07/D07-1073.pdf.

[17] À Bravo, M Cases, N Queralt-Rosinach, F Sanz, and LI Furlong. A knowledge-driven approach to extract disease-related biomarkers from the literature. *BioMed research international*, 2014, 2014.

[18] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: High-performance learning name-finder. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

[19] Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Nyu: Description of the mene named entity system as used in muc-7. In *In Proceedings of the Seventh Message Understanding Conference (MUC-7*, 1998.

[20] Oliver Bender, Franz Josef Och, and Hermann Ney. Maximum entropy models for named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 148–151. Association for Computational Linguistics, 2003.

[21] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 160–163, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119199. URL http://dx.doi.org/10.3115/1119176.1119199.

[22] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119206. URL http://dx.doi.org/10.3115/1119176.1119206.

[23] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *In Proc. 2007 Joint Conference on EMNLP and CNLL*, pages 708–716, 2007.

[24] Tom M Mitchell and Thomas Michell. Machine learning (mcgraw-hill series in computer science), 1997.

[25] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

[26] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics, 2004.

[27] Jian Su, Guodong Zhou, and Guodong Zhou. Named entity recognition using an hmm-based chunk tagger, 2002.

[28] James Mayfield, Paul McNamee, and Christine Piatko. Named entity recognition using hundreds of thousands of features. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL

'03, pages 184–187, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119205. URL http://dx.doi.org/10.3115/1119176.1119205.

[29] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110. Citeseer, 1999.

[30] Sabine Buchholz and Antal van den Bosch. Integrating seed names and ngrams for a named entity list and classifier. In *LREC*. European Language Resources Association, 2000. ISBN 2-9517408-6-7. URL http://dblp.uni-trier.de/db/conf/lrec/lrec2000.html#BuchholzB00.

[31] David J Hand and Keming Yu. Idiot's bayesnot so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

[32] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, pages 301–312. IBM Corp., 2011.

[33] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989.

[34] G David Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[35] John L Fan. Forward-backward algorithm. In *Constrained Coding and Soft Iterative Decoding*, pages 97–116. Springer, 2001.

[36] Thomas G Dietterich. Machine learning for sequential data: A review. In *Structural, syntactic, and statistical pattern recognition*, pages 15–30. Springer, 2002.

[37] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[38] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 168–171, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119201. URL http://dx.doi.org/10.3115/1119176.1119201.

[39] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

[40] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[41] SVN Vishwanathan, Nicol N Schraudolph, Mark W Schmidt, and Kevin P Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, pages 969–976. ACM, 2006.

[42] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[43] Ying He and Mehmet Kayaalp. Biological entity recognition with conditional random fields. In *AMIA Annual Symposium Proceedings*, volume 2008, page 293. American Medical Informatics Association, 2008.

[44] Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics*, 6(Suppl 1):S6, 2005.

[45] Jun Zhao and Feifan Liu. Product named entity recognition in chinese text. *Language Resources and Evaluation*, 42(2):197–217, 2008.

[46] Shaolei Feng, R. Manmatha, and Andrew Mccallum. Exploring the use of conditional random field models and hmms for historical handwritten document recognition. In *the Proceedings of the 2nd IEEE International Conference on Document Image Analysis for Libraries (DIAL*, pages 30–37.

[47] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[48] Alex Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[49] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[50] Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342. Citeseer, 2004.

[51] Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285. ACM, 1988.

[52] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[53] Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

[54] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[55] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JOURNAL OF MACHINE LEARNING RESEARCH*, 3:1137–1155, 2003.

[56] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995. ISBN 0198538642.

[57] Holger Schwenk. Efficient training of large neural networks for language modeling. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 3059–3064. IEEE, 2004.

[58] Holger Schwenk and Jean-Luc Gauvain. Training neural network language models on very large corpora. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 201–208. Association for Computational Linguistics, 2005.

[59] Tomek Strzalkowski, Fang Lin, Jin Wang, and Jose Perez-Carballo. Evaluating natural language processing techniques in information retrieval. In *Natural language information retrieval*, pages 113–145. Springer, 1999.

[60] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query segmentation for web search. In *WWW (Posters)*, 2003.

[61] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th international conference on World Wide Web*, pages 347–356. ACM, 2008.

[62] Jianfeng Gao and Jian-Yun Nie. A study of statistical models for query translation: finding a good unit of translation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201. ACM, 2006.

[63] Erika F De Lima and Jan O Pedersen. Phrase recognition and expansion for short, precision-biased queries based on a query log. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–152. ACM, 1999.

[64] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 391–400. ACM, 2005.

[65] Daniel E Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19. ACM, 2004.

[66] Steven M Beitzel, Eric C Jensen, Ophir Frieder, David Grossman, David D Lewis, Abdur Chowdhury, and Aleksandr Kolcz. Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 581–582. ACM, 2005.

[67] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138. ACM, 2006.

[68] Ying Li, Zijian Zheng, and Honghua Kathy Dai. Kdd cup-2005 report: Facing a great challenge. *ACM SIGKDD Explorations Newsletter*, 7(2):91–99, 2005.

[69] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10. ACM, 2009.

[70] Andrei Z Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238. ACM, 2007.

[71] Steven M Beitzel, Eric C Jensen, Ophir Frieder, David D Lewis, Abdur Chowdhury, and Aleksander Kolcz. Improving automatic query classification via semi-supervised

learning. In *Data Mining, Fifth IEEE international Conference on*, pages 8–pp. IEEE, 2005.

[72] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM, 2008.

[73] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2009.

[74] Satoshi Sekine and Elisabete Ranchhod. *Named entities: recognition, classification and use*, volume 19. John Benjamins Publishing, 2009.

[75] David D. McDonald. Corpus processing for lexical acquisition. chapter Internal and External Evidence in the Identification and Semantic Categorization of Proper Names, pages 21–39. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-02392-X. URL http://dl.acm.org/citation.cfm?id=265994.265996.

[76] Luiz S Oliveira, Robert Sabourin, Flavio Bortolozzi, and Ching Y Suen. A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(06):903–929, 2003.

[77] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[78] GuoDong Zhou, Dan Shen, Jie Zhang, Jian Su, and SoonHeng Tan. Recognition of protein/gene names from text using an ensemble of classifiers. *BMC bioinformatics*, 6(Suppl 1):S7, 2005.

[79] Douglas L Vail, John D Lafferty, and Manuela M Veloso. Feature selection in conditional random fields for activity recognition. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3379–3384. IEEE, 2007.

[80] Maksim Tkachenko and Andrey Simanovsky. Named entity recognition: Exploring features. In *Proceedings of KONVENS*, volume 2012, pages 118–127, 2012.

[81] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[82] John Blitzer, Kilian Q. Weinberger, Lawrence K. Saul, and O C. N. Pereira. Hierarchical distributed representations for statistical language modeling. In *In Advances in Neural Information Processing Systems 17*. MIT Press, 2004.

[83] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078186.

[84] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[85] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[86] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. Citeseer, 2013.

[87] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[88] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[89] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[90] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[91] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[92] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[93] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975. ISSN 0001-0782. doi: 10.1145/360825.360855. URL http://doi.acm.org/10.1145/360825.360855.

[94] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[95] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[96] Stefan Fritsch, Frauke Guenther, and Maintainer Frauke Guenther. Package neuralnet. *Training of neural network,(1.32)*, 2012.